COORDINATED HIGHWAYS ACTION RESPONSE TEAM
STATE HIGHWAY ADMINISTRATION

R1B2A High Level Design

Contract DBM-9713-NMS
TSR # 9803444
Document # M303-DS-004R0

October 16, 2000

By

Computer Sciences Corporation and PB Farradyne Inc

Revision	Description	Pages Affected	Date
0	Initial Release	All	October 16, 2000

Table of Contents

Table of Figures

1 Introduction

1.1 Purpose

This document describes the high level design of the software for Release 1, Build 2A. This software design includes the requirements for this build in the form of Use Cases. The purpose of this software build is to gather speed data from Remote Traffic Microwave Sensor (RTMS) devices and display the data on the Coordinated Highways Action Response Team (CHART) web site.

1.2 Objectives

The main objective of this design is to provide software developers with a framework in which to provide detailed design and implementation of the software components used to allow speed data to be displayed on the CHART web site in the form of a statewide color coded map.

This design also serves to obtain agreement with the requirements as stated in the Use Cases and the approach given to meet the requirements.

1.3 Scope

This design is limited to release 1, build 2A (R1B2A) of the CHART II / Field Management Station (FMS) system and the components to be used on the CHART web site to acquire data from the CHART II system and display the data on the web site.

1.4 Design Process

Object oriented analysis and design techniques were used in creating this design. As such, much of the design is documented using diagrams that conform to the Unified Modeling Language (UML), a de facto standard for diagramming object-oriented designs.

In addition to being object oriented, this design incorporates distributed object techniques, which allow for great flexibility and scalability of the system. In a distributed object system, objects can be deployed in servers throughout the network. This design addresses the partitioning of object types into specific server applications for this release.

The design process is very iterative: each step can possibly cause changes to previous steps. Listed below is the process that was used to create the work products contained in this document:

- The team met with the clients to gain an understanding of the high level requirements of the system. The outcome of this meeting is recorded in document number M361-MM-019R0, “RTMS Requirements Review Meeting Minutes,” 09/08/00.
- The team created a use case diagram to encompass uses of the system as they were stated in the client meeting. Each use case was documented to capture requirements obtained during the client meeting as well as derived requirements.
- A straw man class diagram was created with major entities evident in the use cases being listed as possible classes in the system. High level relationships between the classes were discovered and documented on the class diagram.

- Sequence diagrams were created for each use case, showing at a high level how the classes on the class diagram would be used to perform the use case. This often involved changes to the class diagram, such as adding classes, moving responsibilities between classes, or adding operations to a class. Sometimes the changes affected other sequence diagrams as well.
- After the process of creating sequence diagrams and associated changes to the class diagram, an internal review was held to gain agreement on the design approach and to resolve remaining issues. Minutes from this meeting are contained in document number M361-MM-021R0, "RTMS Coordination Meeting Minutes," 09/20/00.
- The design was broken down into packages, grouping classes with a high amount of dependency together.
- A deployment diagram was created to show the planned deployment of the system.

1.5 Design Tools

The work products contained within this design are extracted from the COOL:JEX design tool. Within this tool, the design is contained in the Chart II project, R1B2 configuration, Analysis phase, system version RTMS.

1.6 Work Products

This design contains the following work products:

- A UML Use Case diagram which captures the requirements of the system.
- A UML Class diagram, showing the high level software objects which will allow the system to accommodate the uses of the system described in the Use Case diagrams.
- UML Sequence diagrams showing how the classes interact to accomplish each use case.
- A UML Package diagram, showing how the classes are broken up into manageable software packages.
- A UML Deployment diagram, showing which servers will serve each class of objects.

2 Software Architecture

The architecture is based around the R1B2 CHART II and FMS architecture. As such, the Common Object Request Broker Architecture (CORBA) is used as the base architecture, with custom built software objects made available on the network to allow their data to be accessed via well defined CORBA interfaces.

The sections below discuss specific elements of the architecture and software components that comprise the system.

2.1 Transportation Sensor Systems

The National Transportation Communications for ITS Protocol (NTCIP) has defined the term Transportation Sensor System (TSS) to describe any system capable of sensing and communicating traffic parameters using the NTCIP protocol. Although the RTMS does not communicate using NTCIP, it otherwise fits into the definition of a Transportation Sensor System. For this reason, the NTCIP terminology is used throughout this design to make the design re-usable for other types of sensors and not be "hard coded" for the type, make, and model sensor being used to provide traffic parameters. Users of the software objects specified in this design, such as the CHART web site, can interface generically with Transportation Sensor System objects to allow for the support of other types of sensors that may be brought online to the CHART II system in the future.

2.2 RTMS Service

The RTMS Service is a standard CHART II service application that serves RTMS software objects, which are on type of Transportation Sensor System. The RTMS Service makes use of the existing CHART II service application framework and benefits from functionality provided by the framework, such as object publication, event based notification, and common startup and shutdown sequences. In addition, the availability of the RTMS Service can be tracked in the CHART II system along with other application services.

An RTMS software object exists in the system for each RTMS in the field that is to provide data to the system. The RTMS Service publishes these objects in the CORBA trading service to allow other applications to discover and use these objects. Each of these RTMS software objects periodically poll their field counterpart and store the current status reported by the field device. Each RTMS object contains a number of changeable configuration values such as a user friendly name for the device, communications parameters, polling frequency, and others.

When there is a change to the status or configuration of an RTMS, an asynchronous event is pushed via a CORBA event channel to allow other applications to be notified of the change. Events are also pushed on an event channel when an RTMS is added to or removed from the system. Changes to values of traffic parameters sensed by an RTMS do not cause an asynchronous event to be pushed due to the dynamic nature of these values. Instead of pushing when there is a change to these values, a periodic push of the current status of all RTMS devices is done.

2.3 TSS Web Client

The TSSClient is an application that runs on the web server and acts as an interface between the CHART II system and the CHART web map. The TSSClient uses CORBA to interface with the CHART II system and passes data received from CHART II to the web map via a database table.

The TSSClient discovers CHART II RTMS objects and event channels that exist in the system by querying the CORBA trader. Upon startup, it asks each RTMS object for its current status and updates the web database with the appropriate information. The TSSClient then listens to events that are pushed from CHART II through CORBA event channels and updates the web database as needed.

2.4 Internet Map Server

The Map Server is used to serve requests from the web server to generate a map. When the web server receives a request for the web page that contains the current speed data, it asks the map server to generate a map image. The map server generates a map image and includes symbols that represent instrumented sections of roadway. The map server queries the web database to retrieve information for each of these symbols and uses the information to color the symbols based on the current average speed for that section of roadway.

2.5 Field Communications

R1B2A uses the communications components designed and developed under FMS R1B2 to communicate with RTMS devices. Refer to *FMS R1B2 High Level Design* for more information on the FMS subsystem. The sections below discuss how the FMS components are used in R1B2A.

2.5.1 Communications Servers (FMS Remote PC)

Communications servers are used in R1B2A to connect to RTMS devices deployed throughout the state of Maryland. A communication server is a PC that is outfitted with one or more pieces of communications hardware, such as Integrated Services Digital Network (ISDN) and Plain Old Telephone System (POTS) modems. This communications hardware is used remotely by the RTMS Service to connect to RTMS devices from a location on the statewide network that is physically close to the device that usually offers reduced communications costs.

2.5.2 Port Manager

A Port Manager is a software object that manages access to the communications hardware on a Communications Server. The RTMS software object acquires communications ports from one or more Port Manager objects. RTMS software objects use a communication port to exchange data with RTMS devices.

2.5.3 RTMS Protocol Handler

The RTMS Protocol Handler is a utility class that encapsulates the communication protocol used to retrieve data from an RTMS device. After a port is retrieved from a Port Manager and

connected to the device, the RTMS Protocol handler is used to send the correct sequence of bytes to the device and interpret the response from the device.

2.6 Database Usage

The CHART II database is used to store configuration data for each RTMS currently known to the system. Data is retrieved from the database at startup to allow the RTMS objects to assume their last known configuration. When configuration values are changed, the data is written immediately to the database so it is available if the server should be restarted. The only status data persisted to the database is the communications mode of the RTMS (online, offline, maintenance mode) and the operational status (OK, COMM_FAILED, and HARDWARE_FAILED). Status data such as speed, volume, and occupancy are absent from the RTMS status until the RTMS is polled for the first time.

The CHART web site uses its own database engine. The TSSClient updates a table in the web database as data from CHART II is received. The table is queried each time a web user requests the web page that contains the map of current speeds.

2.7 ITS National Standards Approach

Components to be developed as a part of R1B2A are designed to be compliant with the current Intelligent Transportation System (ITS) national standards in both the Center-to-Center and Center-to-Field requirements. The Center-to-Center requirements are met because the CORBA interface used to obtain RTMS data is CORBA, one of two methods approved by the National Transportation Communications for ITS Protocols (NTCIP) Center-to-Center committee for communication between ITS software components.

Center-to-Field standards for sensors such as the RTMS have not yet been approved and therefore by default this design meets the current NTCIP Center-to-Field standards. Because of this lack of standards, the communications to the RTMS device uses the manufacturer protocol, encapsulated in the RTMSProtocolHdlr object in this design. Should an NTCIP compliant device exist in the future, a protocol handler that performs NTCIP communications would be written and used in this design where RTMSProtocolHdlr is now used.

Although an approved Center-to-Field standard does not exist, the design takes into account work in progress on the NTCIP standard for Transportation Sensor Systems in its naming and terminology.

3 Models

The following sections provide models and diagrams that show the high level design of the R1B2A software. This section contains a use case diagram for each use of the system, with each use case providing requirements of the system. A class diagram shows software objects that comprise the system and the relationships between them. Sequence diagrams are provided to show how objects interact to accomplish specific uses of the system.

3.1 Use Case Diagrams

3.1.1 RTMSUseCase (Use Case Diagram)

This diagram shows the uses of the “Real Time Traffic Maps” portion of the CHART web site. The main purpose of this effort is to provide graphical display of the status of speed sensed by RTMS devices to CHART Intra/Internet users. Upon web browser request, the web map server will generate a map image, along with associated information of each sensor that will display arrows with different color or shape to show the current average speed being retrieved from CHART system. For example, a sensor monitoring a roadway where the current average speed is between 0 and 29 MPH might be displayed as a red arrow pointing in the direction of the traffic. A roadway with a current average speed between 30 and 54 MPH might be displayed as a yellow arrow pointing in the direction of traffic, etc.

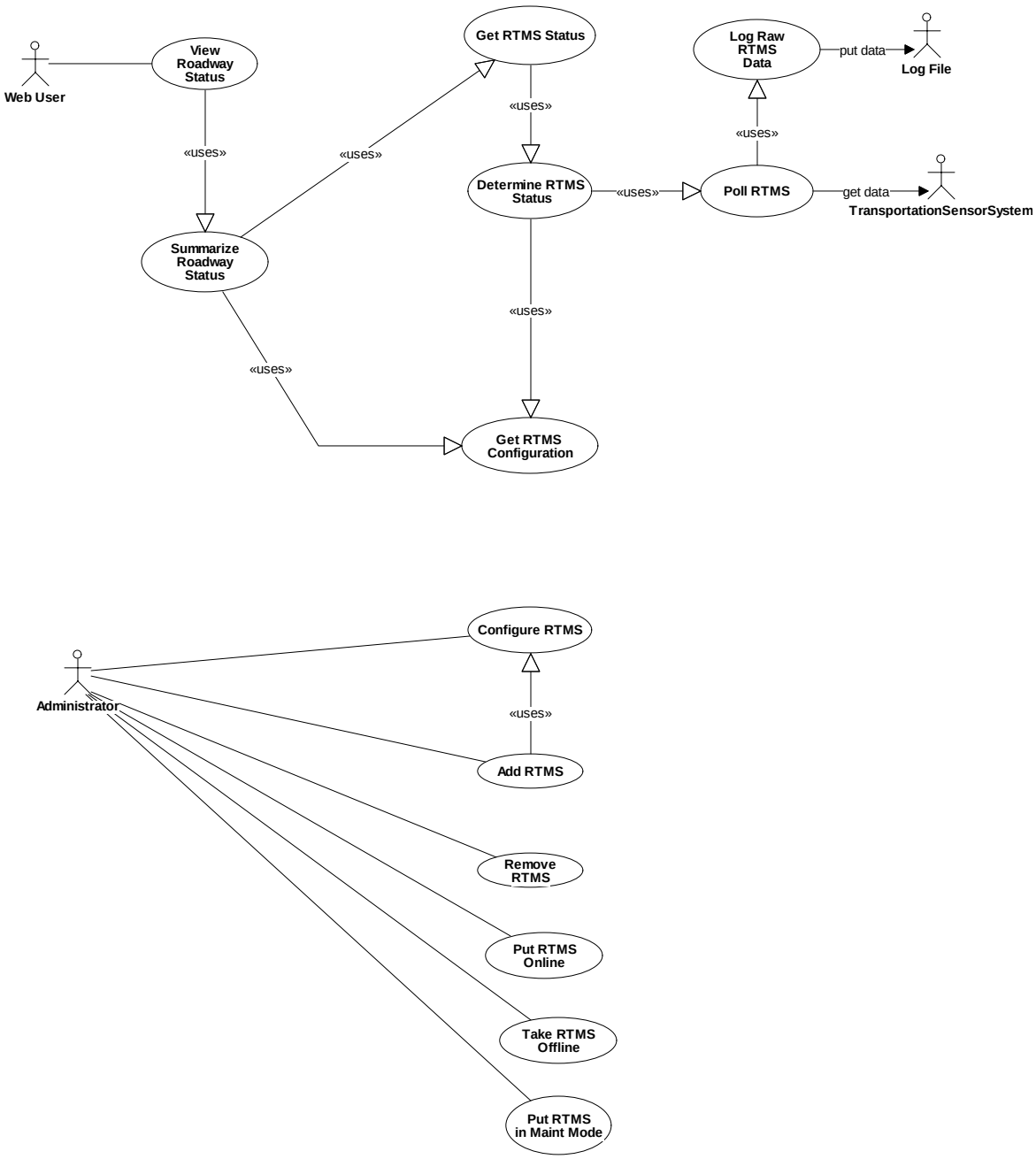

Figure 1. RTMSUseCase (Use Case Diagram)

3.1.1.1 Add RTMS (Use Case)

A CHART II user with the appropriate functional rights (usually an administrative user) shall be able to add an RTMS to the system. During the process of adding the RTMS, the user must configure the RTMS for use in the system. See the Configure RTMS use case for details. When an RTMS is added, it is added in the offline state. It must be placed online before it begins providing data to the system. See the Put RTMS Online use case for details. Notification of the addition of the RTMS is sent to interested parties.

3.1.1.2 Configure RTMS (Use Case)

An RTMS must be configured within the system to match the setup of the device in the field. Each of the eight detection zones of the RTMS that are configured to sense traffic must be placed into logical groupings. A direction must be assigned to each zone group and optionally, a text description of the group may be entered.

In addition to the configuration of the zones of the RTMS into groups, the system shall allow a name to be assigned to each RTMS as well as a textual description of the RTMS location.

Communication parameters shall be able to be configured, which includes the primary and secondary communications server host names, the phone numbers (per comm server), type of modem used to dial the RTMS, the baud rate, data bits, stop bits, parity, and flow control.

The polling rate for the RTMS shall be able to be configured, with a resolution of 1 second and a range from 1 second to 24 hours. The default value shall be 5 minutes. The user must take into consideration the Message Period programmed into the RTMS device when setting the polling rate.

Each RTMS in the system shall be assigned a unique identifier when it is added to the system. This unique identifier shall become part of the RTMS configuration, however it cannot be changed.

3.1.1.3 Determine RTMS Status (Use Case)

The system shall periodically poll each RTMS in the system and determine its current status. The current status of all RTMS objects shall be periodically sent to the web site. The interval on which the current is sent to the web site shall be configurable.

3.1.1.4 Get RTMS Status (Use Case)

The current status for an RTMS may be retrieved. This status shall include the current traffic parameters (volume, speed, occupancy) for each zone group configured for the RTMS and the current mode of the RTMS (online / offline / maintenance mode). Also included is the operational status of the device. This status shall be set to OK if the device was polled successfully on the last attempt. The operational status shall be set to

COMM_FAILURE if the device could not be connected or did not respond during the last poll attempt. The operational status shall be set to HARDWARE_FAILURE if the status returned by the RTMS indicates a problem (byte 11 of the speed message contains a value other than 10, 20, 30, 40, 50, 60, or 70).

When the mode or operational status for an RTMS changes, an event is pushed to allow other software processes to be asynchronously notified of the changes. Changes to values of traffic parameters shall not cause an asynchronous event to be pushed, and instead the current status of all RTMSs in the system (including the current values for traffic parameters) shall be pushed periodically on regular intervals.

3.1.1.5 Get RTMS Configuration (Use Case)

The configuration values as set in the Set RTMS Configuration use case can be retrieved to determine the current configuration of the RTMS. See the Set RTMS Configuration use case for details on specific values that may be obtained.

When a configuration value for an RTMS changes, an event is pushed on an event channel to allow software processes to be asynchronously notified of configuration changes.

3.1.1.6 Log Raw RTMS Data (Use Case)

The raw data for each RTMS shall be logged to a flat file when the RTMS is polled. Each line of data in the log file shall be tagged with the ID and name of the RTMS. The data shall be logged in text format and shall include the volume, occupancy, and speed reported for each of the 8 RTMS detection zones. The data shall only be logged for an RTMS that is in online mode. A new file shall be created and used for each day's data. Old log files shall be deleted automatically. The maximum age of a log file (before it is considered old) shall be configurable.

3.1.1.7 Poll RTMS (Use Case)

The system shall support the polling of the X2 model RTMS. When the RTMS is polled, a connection shall be established via modem and a data request shall be sent to the device. The response from the device shall be checked for format and data errors.

3.1.1.8 Put RTMS in Maint Mode (Use Case)

A user with the proper functional rights (usually assigned to an administrator) shall be able to change the mode of an RTMS from online or offline to maintenance mode. A notification of the mode change shall be sent to interested parties.

When an RTMS is placed into maintenance mode, the system continues to poll the device, however all status (queried or via asynchronous notification) shall indicate that the device is in maintenance mode. Additionally, raw data shall not be logged for an RTMS in maintenance mode.

3.1.1.9 Put RTMS Online (Use Case)

A user with the proper functional rights (usually assigned to an administrator) shall be able to change the mode of an RTMS from offline or maintenance mode to online mode. Placing an RTMS online shall start the polling of the RTMS (if it was offline) and make its data eligible for use by clients such as the web site. A notification of the mode change shall be sent to interested parties.

3.1.1.10 Remove RTMS (Use Case)

A user with the proper functional rights (usually assigned to an administrator) shall be able to remove an RTMS from the system. When an RTMS is removed, notification of its removal shall be sent to interested parties.

3.1.1.11 Summarize Roadway Status (Use Case)

The status of all individual RTMS devices is collected and provided in a format that can be used to satisfy web requests for the current roadway status.

3.1.1.12 Take RTMS Offline (Use Case)

A user with the proper functional rights (usually assigned to an administrator) shall be able to change the mode of an RTMS from online or maintenance mode to offline. When an RTMS is placed offline, the system shall cease polling of the device. A notification of the mode change shall be sent to interested parties.

3.1.1.13 View Roadway Status (Use Case)

The current status of Maryland's roadways that are equipped with RTMS devices can be viewed in the form of a map, with colors and symbols used to represent the current speed range at a given point on the road.

3.2 Class Diagrams

3.2.1 TransportationSensorSystem (Class Diagram)

This class diagram shows software objects and interfaces that provide an interface to RTMS data contained within the CHART II system. While the current requirement is to retrieve data from RTMS devices, this design uses terminology and design concepts that apply to other sensor devices known in the NTCIP impending standard as Transportation Sensor Systems (TSS).

This diagram is focused on the CHART II interfaces that are presented to allow other applications to access TSS data. Implementation specific details are to be included in a detailed design.

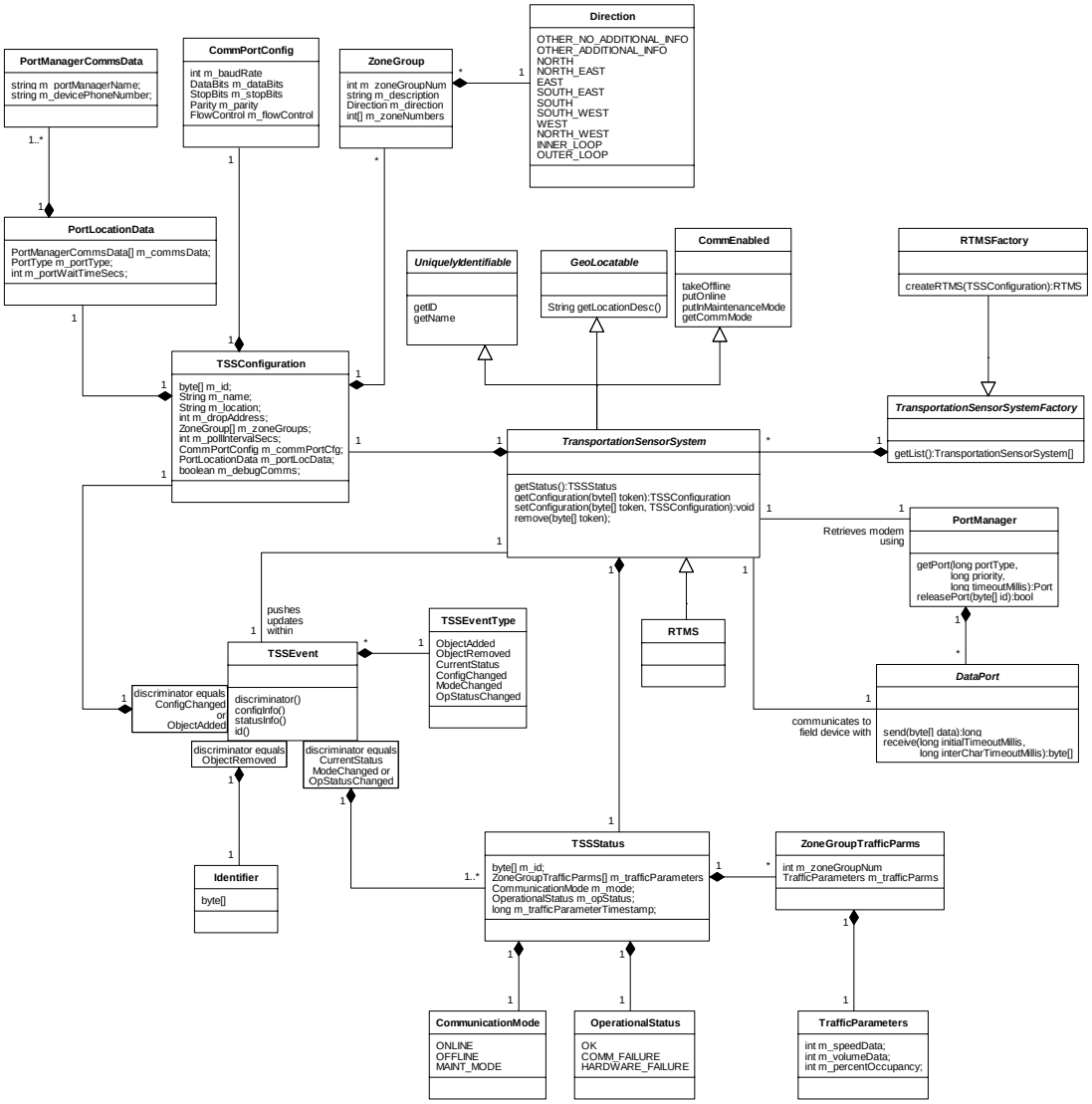

Figure 2. TransportationSensorSystem (Class Diagram)

3.2.1.1 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can have their communications turned on or off. This typically only applies to field devices.

3.2.1.2 CommPortConfig (Class)

This structure is used to pass comm port configuration values during a connection request.

3.2.1.3 CommunicationMode (Class)

The CommunicationMode class enumerates the modes of operation for a device: ONLINE, OFFLINE, and MAINT_MODE. ONLINE is used to indicate the device is available to the operational system. Offline is used to indicate the device is not available to the operational system and communications to the device have been disabled. MAINT_MODE is used to indicate that the device is available only for maintenance / repair activities and testing.

3.2.1.4 DataPort (Class)

A DataPort is a port that allows binary data to be sent and received. Ports of this type support a receive method that allows a chunk of all available data to be received. This method prevents callers from having to issue many receive calls to parse a device response. Instead, this *receive call* returns all available data received within the timeout parameters. The caller can then parse the data on their side. Using this mechanism, device command and response should require only one call to send and one call to receive.

3.2.1.5 Direction (Class)

This enumeration defines direction of travel.

3.2.1.6 GeoLocatable (Class)

This interface must be supported by any system object that can be located via a geographic reference. This interface will be expanded in future releases to include the information necessary for placing objects on a system map.

3.2.1.7 Identifier (Class)

The identifier typedef is used to represent a CHART II unique identifier, which is a byte array containing 32 bytes.

3.2.1.8 OperationalStatus (Class)

The OperationalStatus class enumerates the types of operational status a device can have: OK (normal status), COMM_FAILURE (no communications to the device), or HARDWARE_FAILURE (device is reachable but is reporting a hardware failure).

3.2.1.9 PortLocationData (Class)

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m_commsData – One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m_portType – The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m_portWaitTimeSecs – The maximum number of seconds to wait when attempting to acquire a port from a port manager.

3.2.1.10 PortManager (Class)

A PortManager is a software object that manages access to physical communications ports on a computer. The port manager allows ports to be requested by type and priority. When the demand for a specific type of port is greater than the supply, the PortManager queues the requests using priority. The PortManager also allows a timeout to be specified to indicate the amount of time the caller is willing to wait for a port to become available.

3.2.1.11 PortManagerCommsData (Class)

This class contains values that identify a port manager and the phone number to dial to access a device from the given port manager. This class exists to allow for the phone number used to access a device to differ based on the port manager to take into account the physical location of the port manager within the telephone network. For example, when dialing a device from one location the call may be long distance but when dialing from another location the call may be local.

3.2.1.12 RTMS (Class)

A Remote Traffic Microwave Sensor (RTMS) is a type of TransportationSensorSystem manufactured by EIS inc. It senses traffic parameters for eight detection zones. This is a tagging interface used to distinguish RTMS objects from other types of TransporationSensorSystem objects in the system.

3.2.1.13 RTMSFactory (Class)

This interface is implemented by objects that can create and serve TransportationSensorSystem objects that provide access to RTMS devices.

3.2.1.14 TrafficParameters (Class)

This struct contains traffic parameters that are sensed and reported by a Traffic Sensor System such as the RTMS.

m_speedData – The average speed collected over a sample period in miles per hour in tenths (*thus* 550 = 55.0 MPH). Valid values are 0 to 2550. A value of 65535 is used to indicate a missing or invalid value (such as when the volume for the sample period is zero).

m_volumeData – The count of vehicles for the sample period. Valid values 0 to 65535. A value of 65535 represents a missing value.

m_percentOccupancy – The percentage of occupancy of the roadway in tenths of a percent (*thus* 1000 = 100.0 percent). Valid values are 0 to 1000. A value of 65535 represents a missing or invalid value.

3.2.1.15 TransportationSensorSystem (Class)

A Transportation Sensor System (TSS) is a generic term used to describe a class of technology used for detection within the transportation industry. Examples of TSS devices range from the advanced devices, such as RTMS, to basic devices, such as single loop detectors.

This software interface is implemented by objects that provide access to the traffic parameters sensed by a Transportation Sensor System. Transportation Sensor Systems are capable of providing detection for one or more detection zones, known as Sensor Zones. A single loop detector would have one sensor zone, while an RTMS would have eight sensor zones.

3.2.1.16 TransportationSensorSystemFactory (Class)

This interface is implemented by objects that are used to create and serve Sensor System Objects. All factories of sensor system objects can return the list of Sensor System objects, which they have created and serve. Derived interfaces are used to provide factories to create specific make, models, and types of TransportationSensorSystem objects.

3.2.1.17 TSSConfiguration (Class)

This class holds configuration data for a transportation sensor system (TSS) as follows:

m_id – The unique identifier for this sensor system. This field is ignored when the object is passed to the SensorSystem to change its configuration.

m_name – The name used to identify the SensorSystem.

m_location – A descriptive location of the SensorSystem.

m_dropAddress – The drop address for the device.

m_zoneGroups – logical groupings of detection zones, used to provide a single set of traffic parameters for one or more detection zones.

m_pollIntervalSecs – The interval on which the SensorSystem should be polled for its current traffic parameters (in seconds).

m_commPortCfg – Communication configuration values.

m_portLocData – Configuration information that determines which port manager(s) should be used to establish a connection with the SensorSystem.

m_debugComms – Flag used to enable/disable the logging of communications data for this SensorSystem. When enabled, command and response packets exchanged with the device are logged to the application's log file.

3.2.1.18 TSSEvent (Class)

The class is a CORBA union that contains varying data depending on the current value of the discriminator.

If the discriminator is ConfigChanged or ObjectAdded, this union contains a TSSConfig object.

If the discriminator is ObjectRemoved, this union contains a byte[] containing the unique identifier for the Traffic Sensor System that was removed.

If the discriminator is CurrentStatus, the union contains an array of one or more TSSStatus objects.

If the discriminator is ModeChanged, or OpStatusChanged, the union contains a single TSSStatus object.

3.2.1.19 TSSEventType (Class)

This enumeration defines the types of events that may be pushed on an event channel by a Transportation Sensor Status object. The values in this enumeration are used as the discriminator in the TSSEvent union.

ObjectAdded – a TransportationSensorSystem has been added to the system.

ObjectRemoved – a TransportationSensorSystem has been removed from the system.

CurrentStatus – The event contains the current status of one or more Transportation Sensor System objects.

ConfigChanged – One or more configuration values for the Transportation Sensor System has been changed.

ModeChanged – The communications mode of the TransportationSensorSystem has changed.

OpStatusChanged – The operational status of the TransportationSensorSystem has changed.

3.2.1.20 TSSStatus (Class)

This class holds current status information for the RTMS as follows:

m_id – The ID of the RTMS for which this status applies.

m_trafficParameters – The traffic parameters for zone group of the Transporation Sensor System as specified in the Sensor system's TSSConfig object.

m_mode – The communication mode of the TSS.

m_opStatus – The operational status for the TSS.

m_trafficParameterTimestamp – A timestamp that records when the traffic parameter data was collected from the device.

3.2.1.21 UniquelyIdentifiable (Class)

This interface is implemented by classes whose instances have a unique identifier that is guaranteed not to match the identifier of any other uniquely identifiable objects in the system.

3.2.1.22 ZoneGroup (Class)

This class is used to group one or more detection zones of a Transportation Sensor System into a logical grouping. Traffic parameters for all detection zones included in the group are averaged to provide a single set of traffic parameters for the group.

3.2.1.23 ZoneGroupTrafficParms (Class)

This struct contains traffic parameters for a ZoneGroup.

m_zoneGroupNumber – The number of the zone group for which the traffic parameters apply.

m_trafficParms – The traffic parameter values for the zone group.

3.3 Sequence Diagrams

3.3.1 AddRTMS:Basic (Sequence Diagram)

An administrator can add an RTMS to the system. The administrator uses the Graphical User Interface (GUI) to start the add operation. A TSSConfiguration object is created in the GUI and values are changed by the administrator via the properties dialog. When the administrator is finished setting the configuration values, the administrator presses the OK button to cause the RTMS to be created with the specified configuration values.

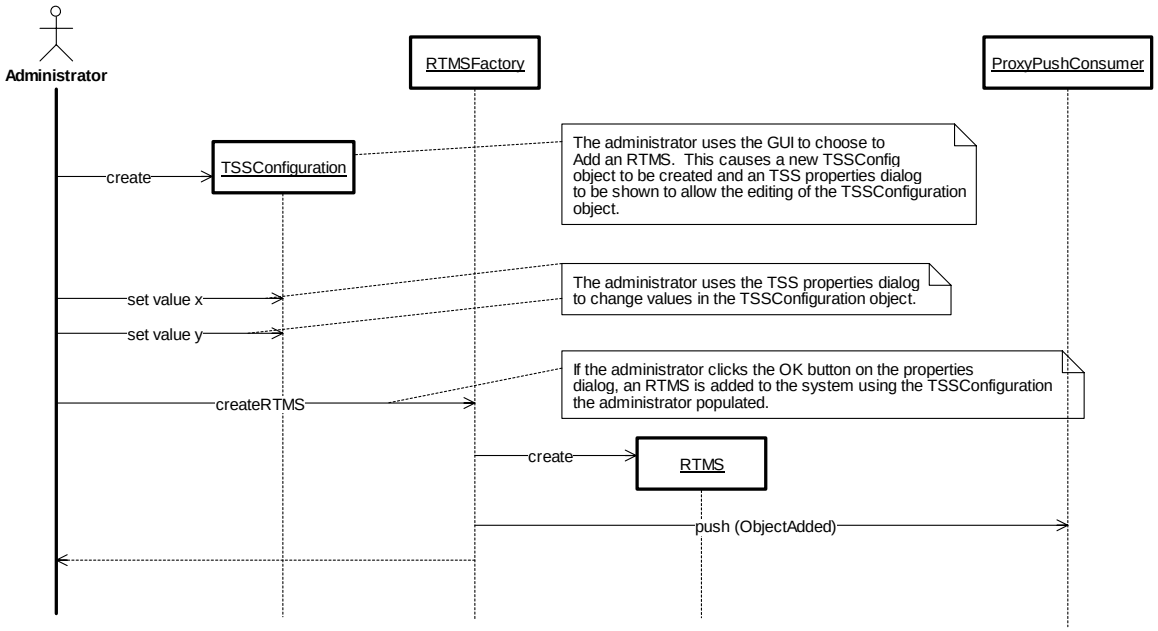

Figure 3. AddRTMS:Basic (Sequence Diagram)

3.3.2 ConfigureRTMS:Basic (Sequence Diagram)

An administrator can change the configuration of an RTMS. A GUI is used to first view the current configuration and then to change any of the configuration values. If the administrator wishes to save the changes, the OK button is pressed to send the new configuration to the RTMS object. If the administrator wishes to cancel without saving changes, they can press the cancel button on the dialog. If a new configuration is set, the RTMS object stores the configuration values and pushes an update to let others know of the configuration change.

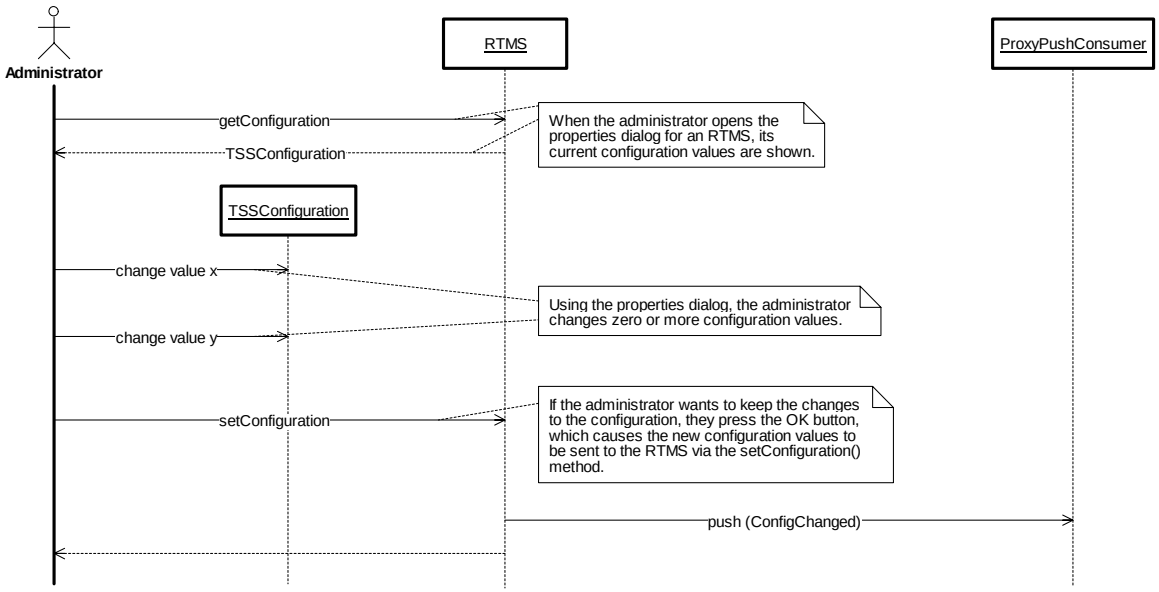

Figure 4. ConfigureRTMS:Basic (Sequence Diagram)

3.3.3 DetermineRTMSStatus:Basic (Sequence Diagram)

The RTMS software object periodically polls the RTMS device and retrieves the traffic parameters. It summarizes the lane (detection zone) level data according to the zone groups that are specifed in the TSSConfiguration object. This summarization is the average speed for all detection zones in the group, the sum of the volume from each detection zone in the group, and the average occupancy of the detection zones in the zone group. If there is no volume for a detection zone, the speed last sensed in the detection zone is used in the summary data in place of the actual speed of zero. (The volume and occupancy reported by the RTMS are used as normal in this case.)

If a change to the operational status occurs due to a communication failure or because the device is reporting a hardware problem, an asynchronous event is pushed to provide notification of the change to other applications.

Note that an asynchronous event with the current traffic parameter data is not pushed during the polling task for the RTMS. Instead, the data for all RTMS devices is pushed periodically. See the DetermineRTMSStatus:CurrentStatusPush diagram for details.

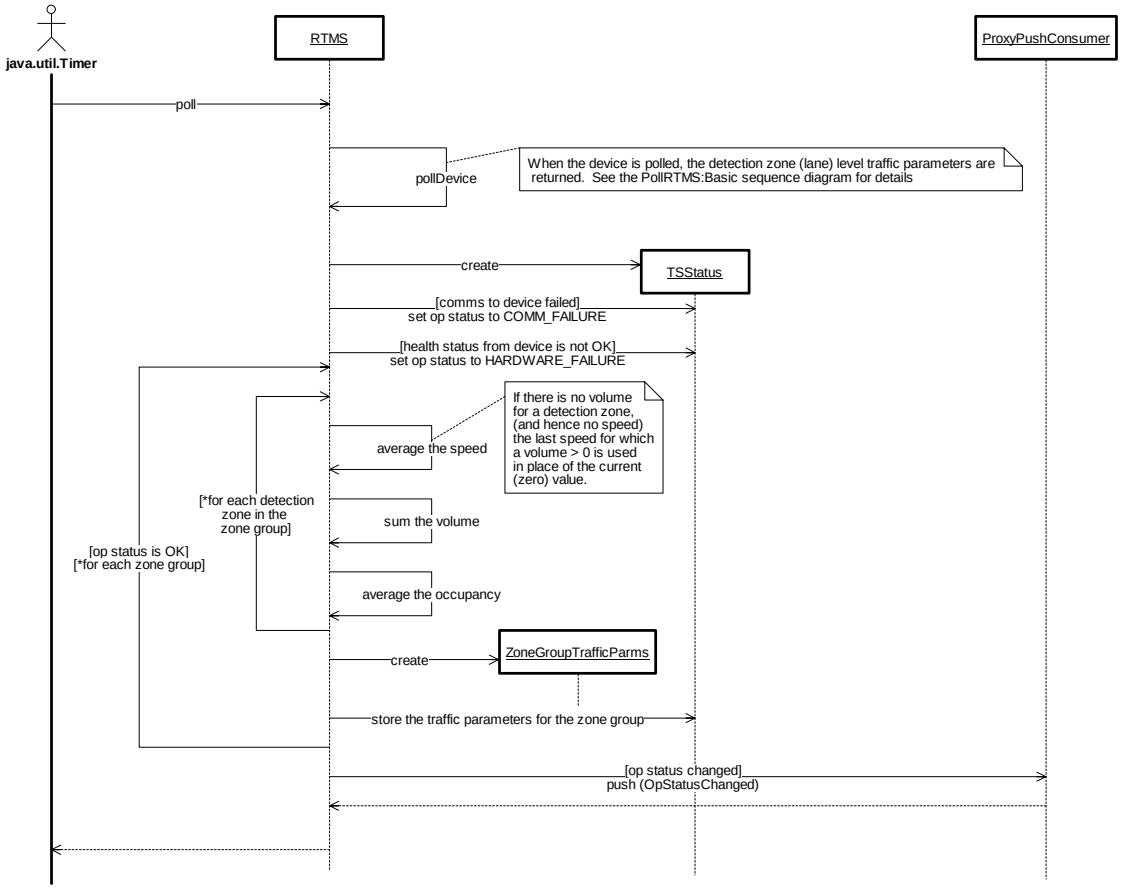

Figure 5. DetermineRTMSStatus:Basic (Sequence Diagram)

3.3.4 DetermineRTMSStatus:CurrentStatusPush (Sequence Diagram)

Because traffic parameter data is very dynamic, the system does not push asynchronous notifications with each change. Instead, the system periodically pushes an asynchronous event with the current status of all RTMS devices.

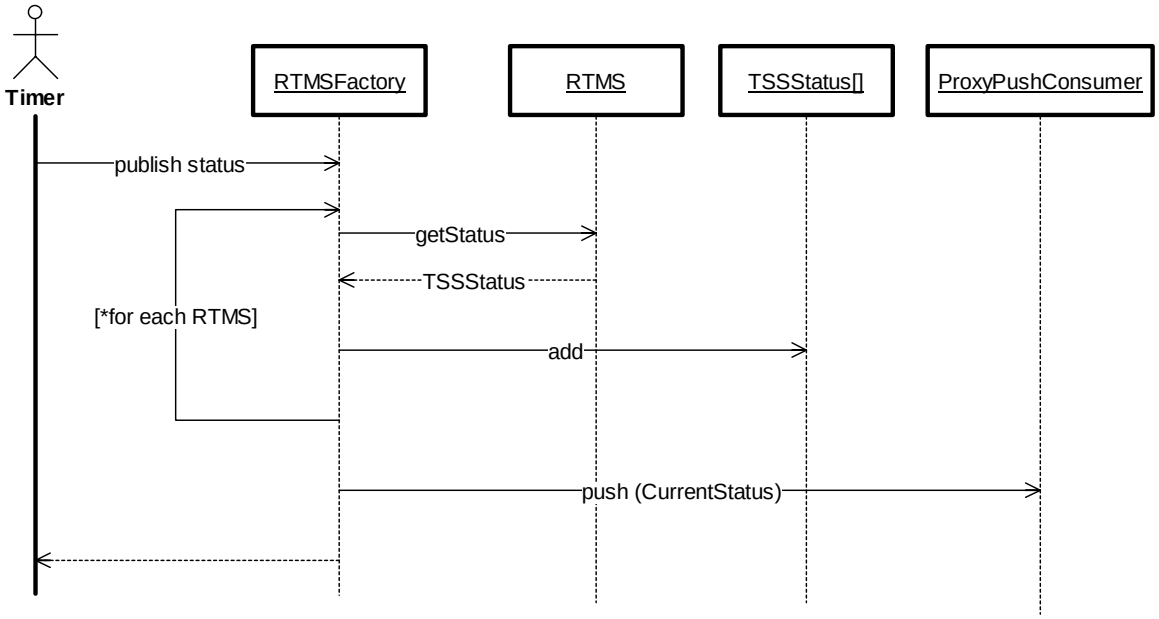

Figure 6. DetermineRTMSStatus:CurrentStatusPush (Sequence Diagram)

3.3.5 GetRTMSConfiguration:Basic (Sequence Diagram)

Each RTMS object maintains its current configuration values as set during pre-deployment population, set via changes to the configuration, or set as the initial values when the object was added to the CHART II system. When asked for its configuration, the RTMS returns the latest configuration values that have been set.

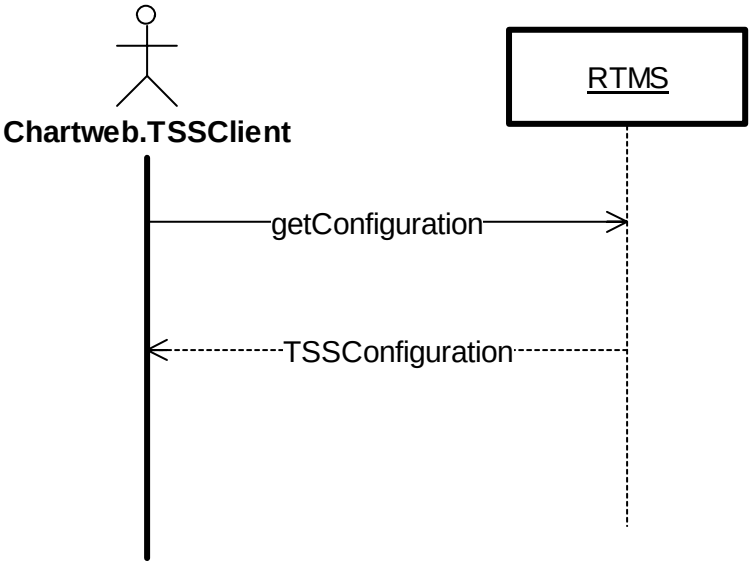

Figure 7. GetRTMSConfiguration:Basic (Sequence Diagram)

3.3.6 GetRTMSStatus:Basic (Sequence Diagram)

TransporationSensorSystem objects maintain their current status as of the last poll of the field device. When asked for the current status, the object returns the TSSStatus with values that were set when the field device was last polled. This status information includes a timestamp to allow one to determine the age of the data.

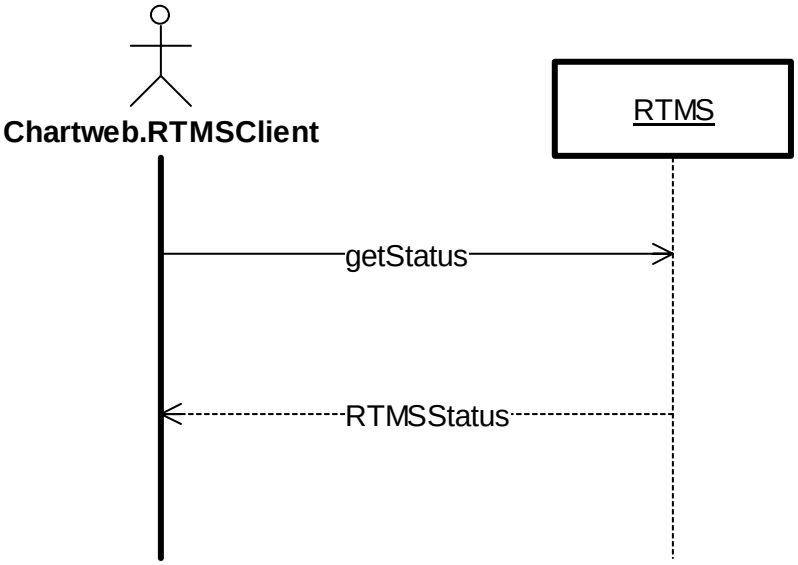

Figure 8. GetRTMSStatus:Basic (Sequence Diagram)

3.3.7 LogRawRTMSData:Basic (Sequence Diagram)

Each time an RTMS is contacted and data is retrieved, the data is logged on a single line in the RTMS raw data text file. Each line is marked with the current date and time as well as identifying information for the RTMS. Each field on a line is separated from the previous field with a comma. The use of a text file makes the data human readable (if necessary), and the use of a comma delimited format allows the data to be read into third party applications or databases easily.

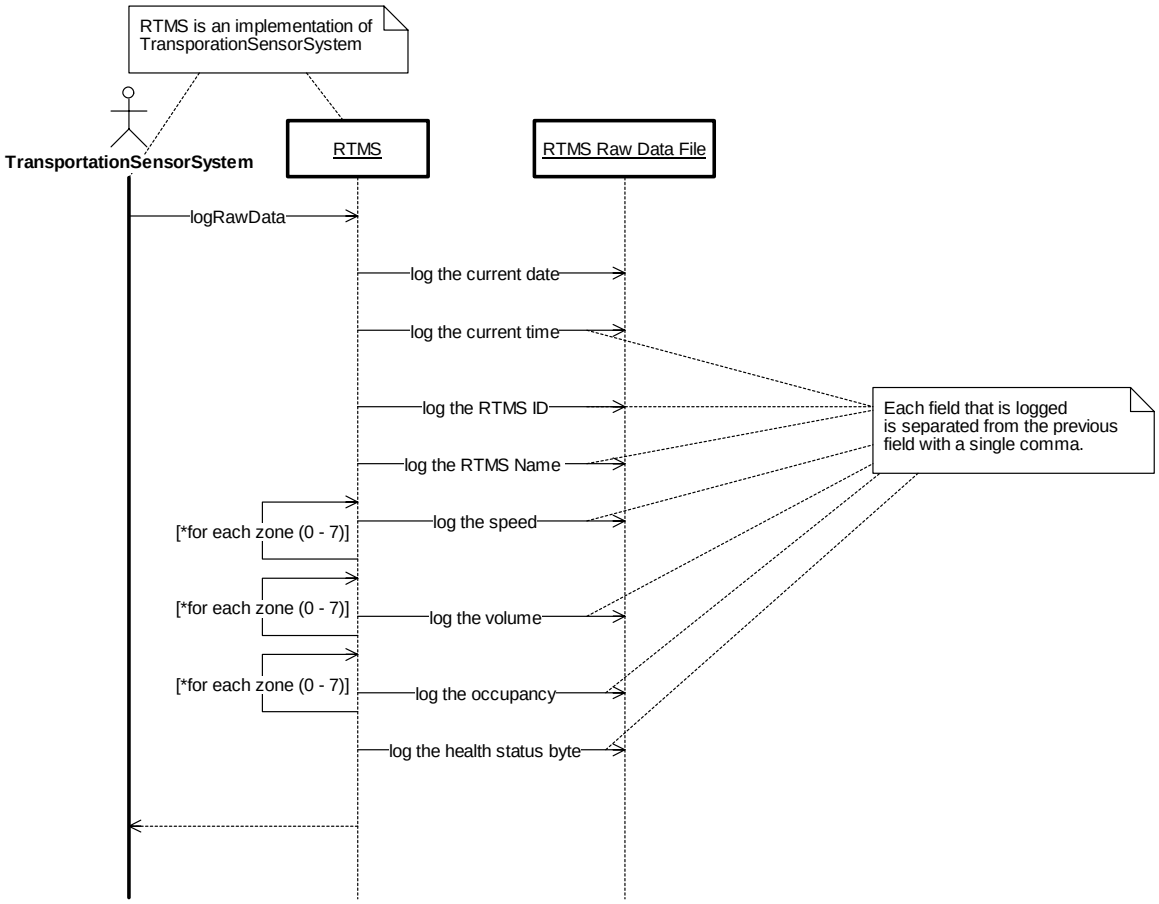

Figure 9. LogRawRTMSData:Basic (Sequence Diagram)

3.3.8 PollRTMS:Basic (Sequence Diagram)

When the RTMS object polls the RTMS device in the field, it obtains access to a modem from a PortManager object and uses the modem to connect to the field device. If a connection to the field device cannot be established, the device is marked as being comm failed and an event is pushed if the operational status was not previously COMM_FAILURE. The device is also marked as COMM_FAILURE if the field device does not respond to the data request command.

If a response is received from the device, the raw data is logged to the RTMS raw data log file in a comma delimited format. The device status field in the response data is checked for an indication of a hardware failure. If a hardware failure is detected, the RTMS is marked as hardware failed and an event is pushed if the operational status was not previously HARDWARE_FAILURE.

If none of the above errors occurred, the zone level data is passed to the caller of pollDevice().

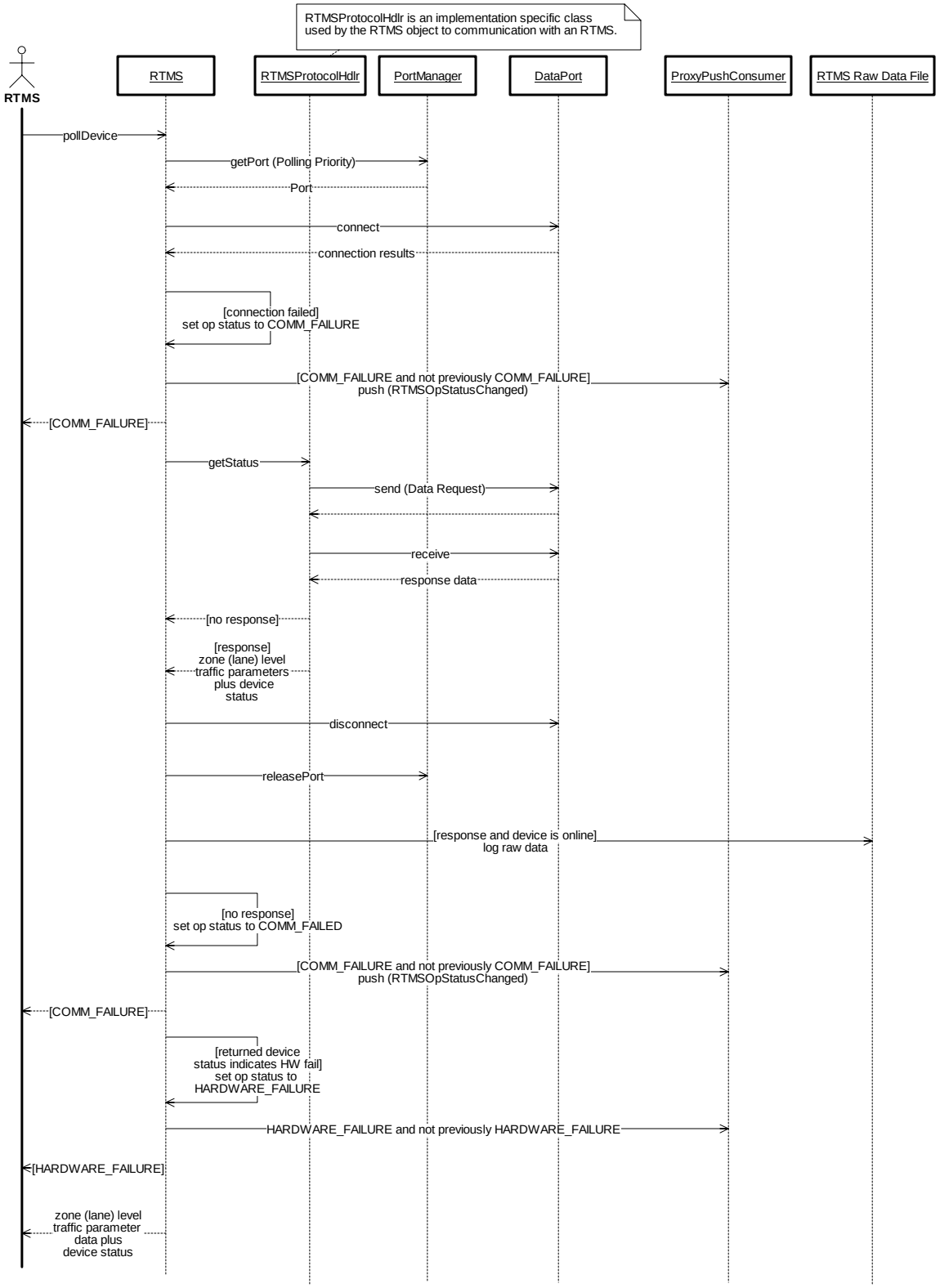

Figure 10. PollRTMS:Basic (Sequence Diagram)

3.3.9 PutRTMSinMaintMode:Basic (Sequence Diagram)

An administrator can put an RTMS into maintenance mode if the RTMS is currently online or offline. When placed into maintenance mode, the raw data logging for the RTMS is disabled. If the RTMS was offline (and therefore polling was disabled), polling is enabled. An event is pushed to notify other applications and objects of the mode change.

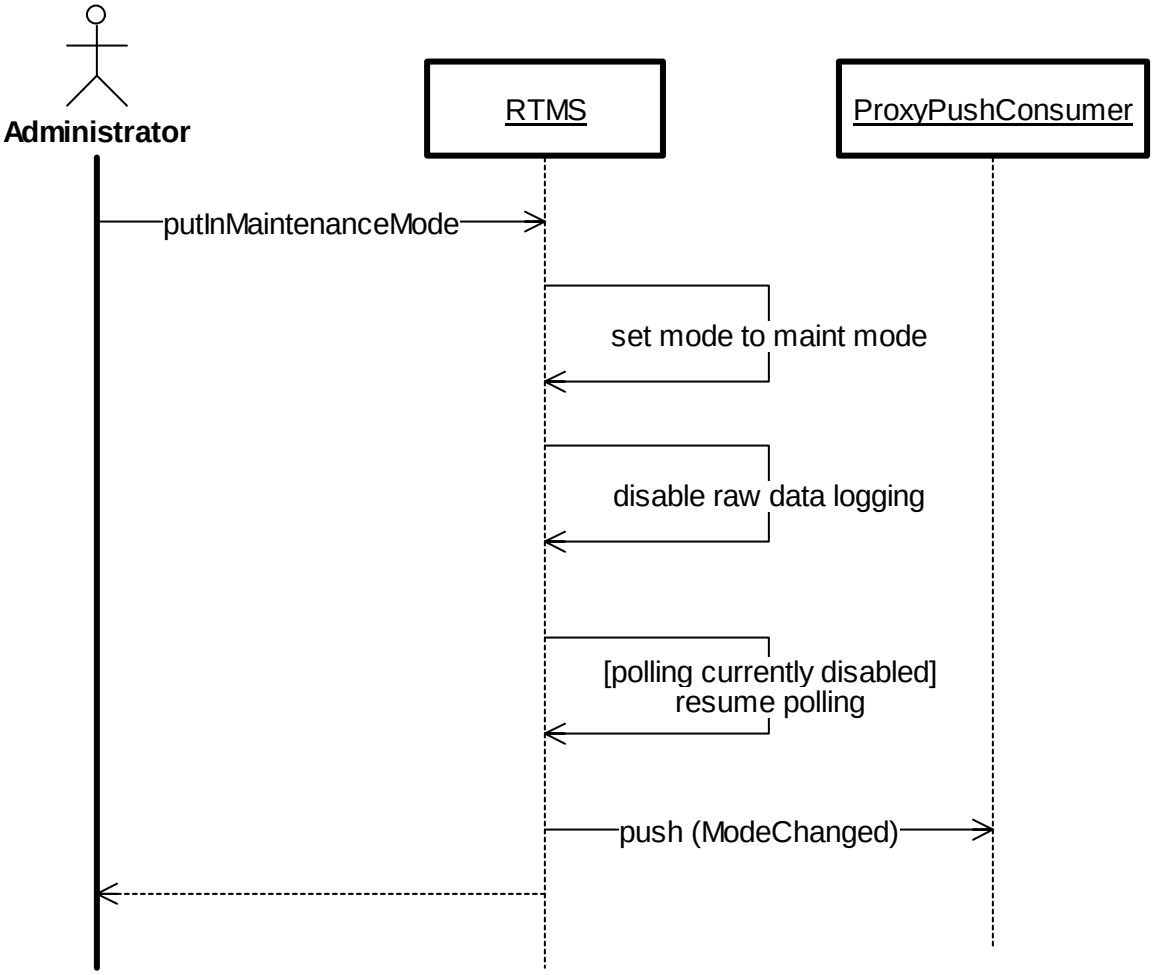

Figure 11. PutRTMSinMaintMode:Basic (Sequence Diagram)

3.3.10 PutRTMSOnline:Basic (Sequence Diagram)

An administrator can put an RTMS online if it is currently in maintenance mode or offline. If the RTMS was offline (and therefore polling was disabled), polling is enabled. Raw data logging is enabled when the RTMS is placed online. An event is pushed to notify other applications and objects of the mode change.

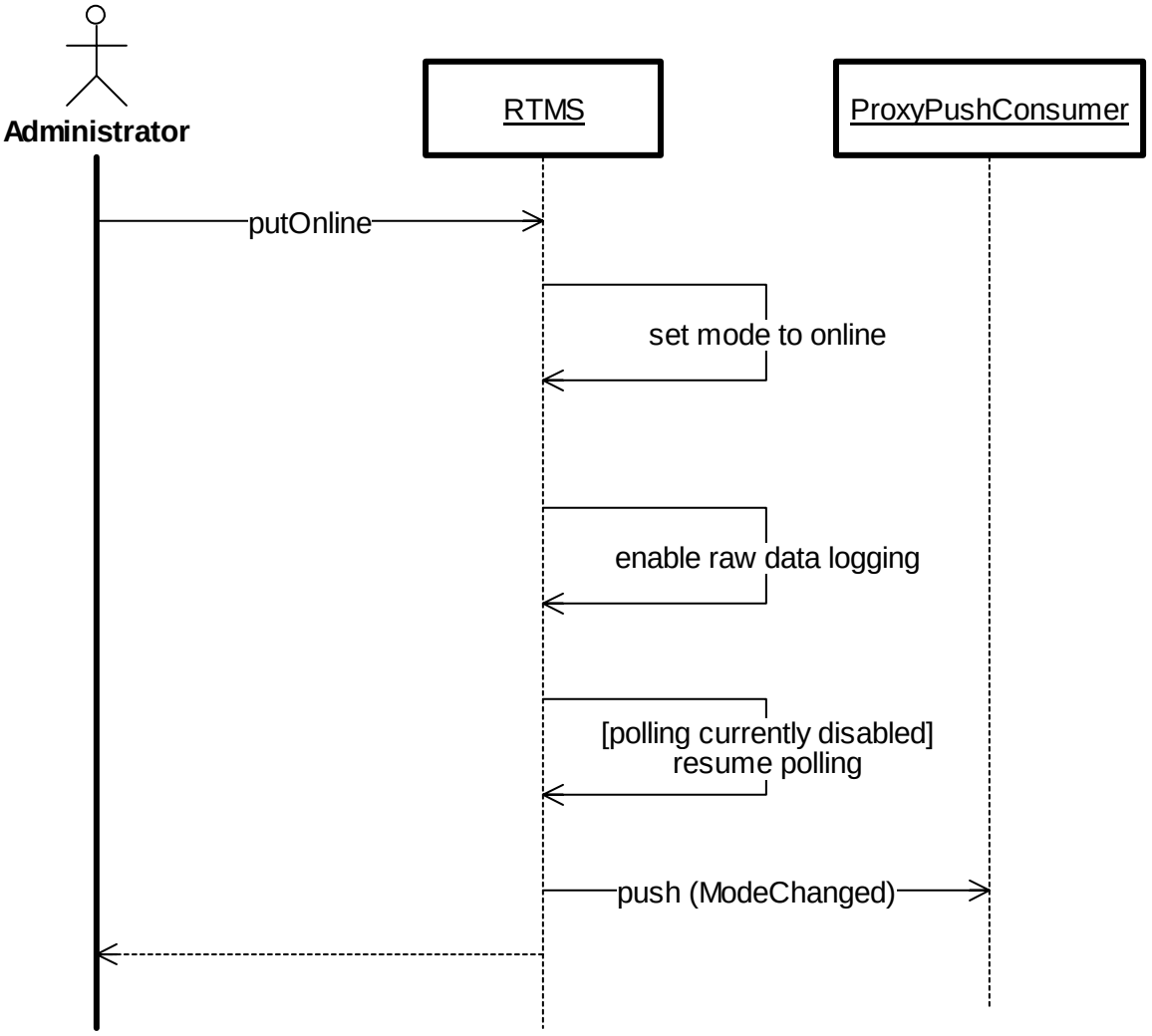

Figure 12. PutRTMSOnline:Basic (Sequence Diagram)

3.3.11 RemoveRTMS:Basic (Sequence Diagram)

An administrator can use the GUI to remove an RTMS from the system. The administrator is provided a warning message and asked to confirm the removal. Once confirmed, the remove method is called on the RTMS, which delegates the work to the RTMSFactory that originally created the RTMS. An event is pushed to notify other applications and objects of the removal of the RTMS.

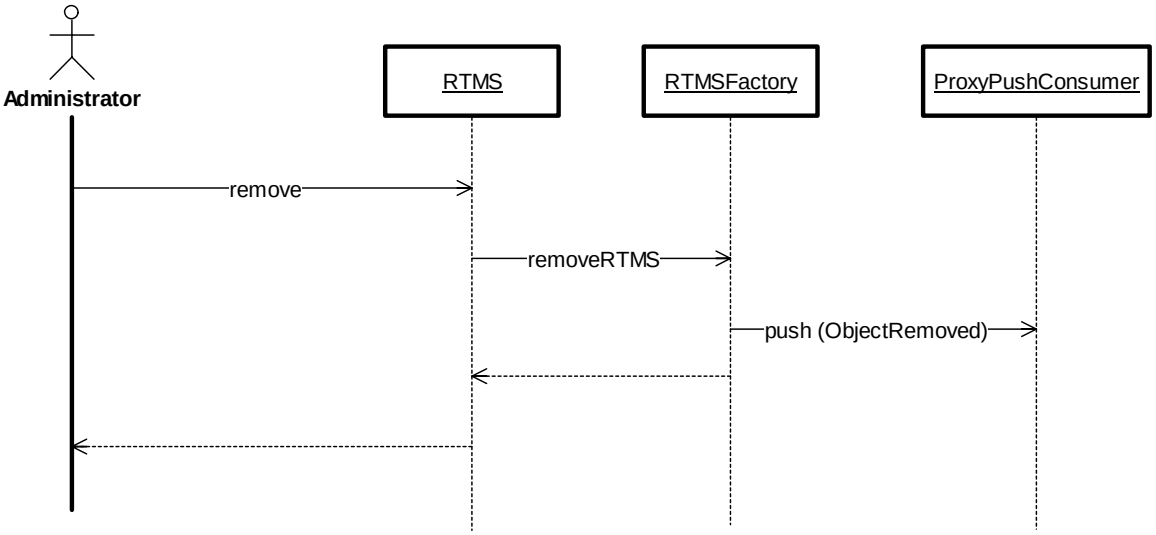

Figure 13. RemoveRTMS:Basic (Sequence Diagram)

3.3.12 SummarizeRoadwayStatus:ConfigChanged (Sequence Diagram)

The CHART II system pushes an event to the Chartweb.TSSClient when the configuration of an RTMS is changed in the CHART II system. The Chartweb.TSSClient handles this event by adding new rows to the web database for each zone group configured for the RTMS if a row does not currently exist. Also, rows that exist for zone groups, which the RTMS configuration no longer contains, are removed from the database.

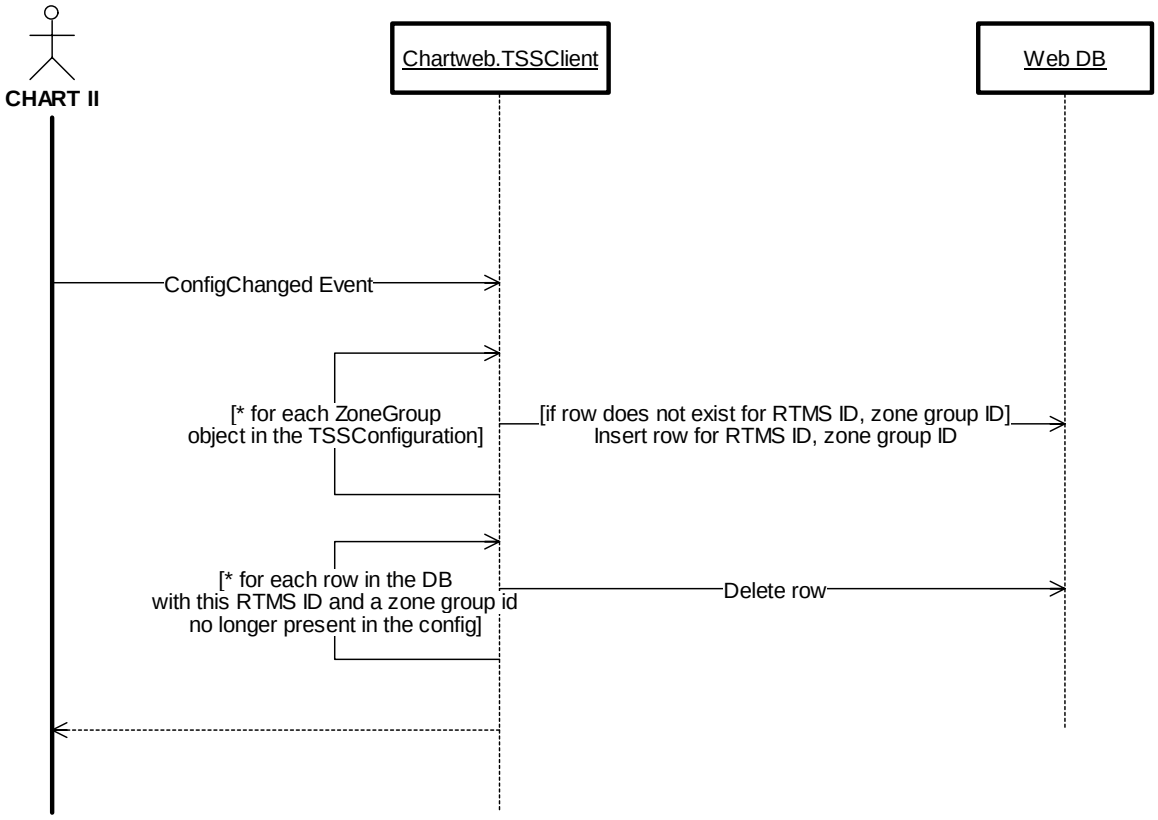

Figure 14. SummarizeRoadwayStatus:ConfigChanged (Sequence Diagram)

3.3.13 SummarizeRoadwayStatus:CurrentStatus (Sequence Diagram)

The CHART II system provides status updates to the Chartweb.TSSClient periodically on a regular interval. The Chartweb.TSSClient handles this event by storing the new speed range for each zone group of each RTMS in the web database, which is keyed on the CHART II ID and zone group ID.

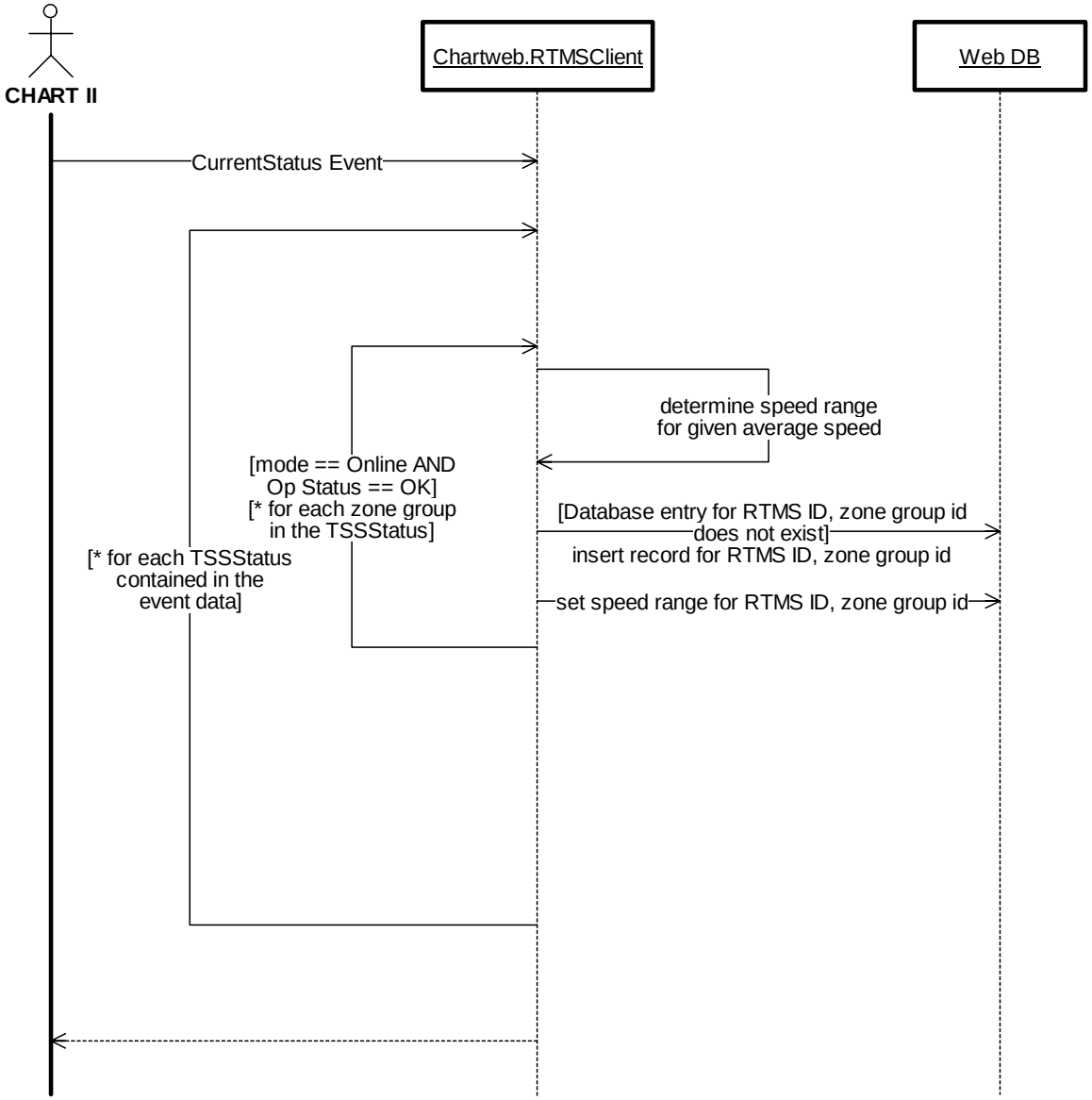

Figure 15. SummarizeRoadwayStatus:CurrentStatus (Sequence Diagram)

3.3.14 SummarizeRoadwayStatus:Initialize (Sequence Diagram)

When the Chartweb.TSSClient is first initialized, it prepares itself to receive asynchronous updates of TSSStatus from the CHART II. Chartweb.TSSClient then gets the current state of RTMS objects from CHART II and sets the web database data to match the CHART II current state. After this initialization is complete, updates to the status of RTMS devices are received asynchrounously as changes occur, at which time the Chartweb.TSSClient makes appropriate updates to the web database. See the other SummarizeRoadwayStatus sequence diagrams for details.

If the Chartweb.TSSClient is unable to contact the CHART II trader or event service, all rows in the web DB are marked offline and this sequence is retried periodically.

If individual RTMS objects within the CHART II system cannot be contacted, rows for the specific RTMS are marked offline in the web database and this sequence is retried periodically.

This sequence is also carried out when the Chartweb.TSSClient suspects the CHART II system has gone down due to the lack of events received from CHART II. When this occurs, this initialization sequence will serve to verify the current status of the RTMS objects or to confirm that CHART II is not fully available and to mark the appropriate RTMS objects offline.

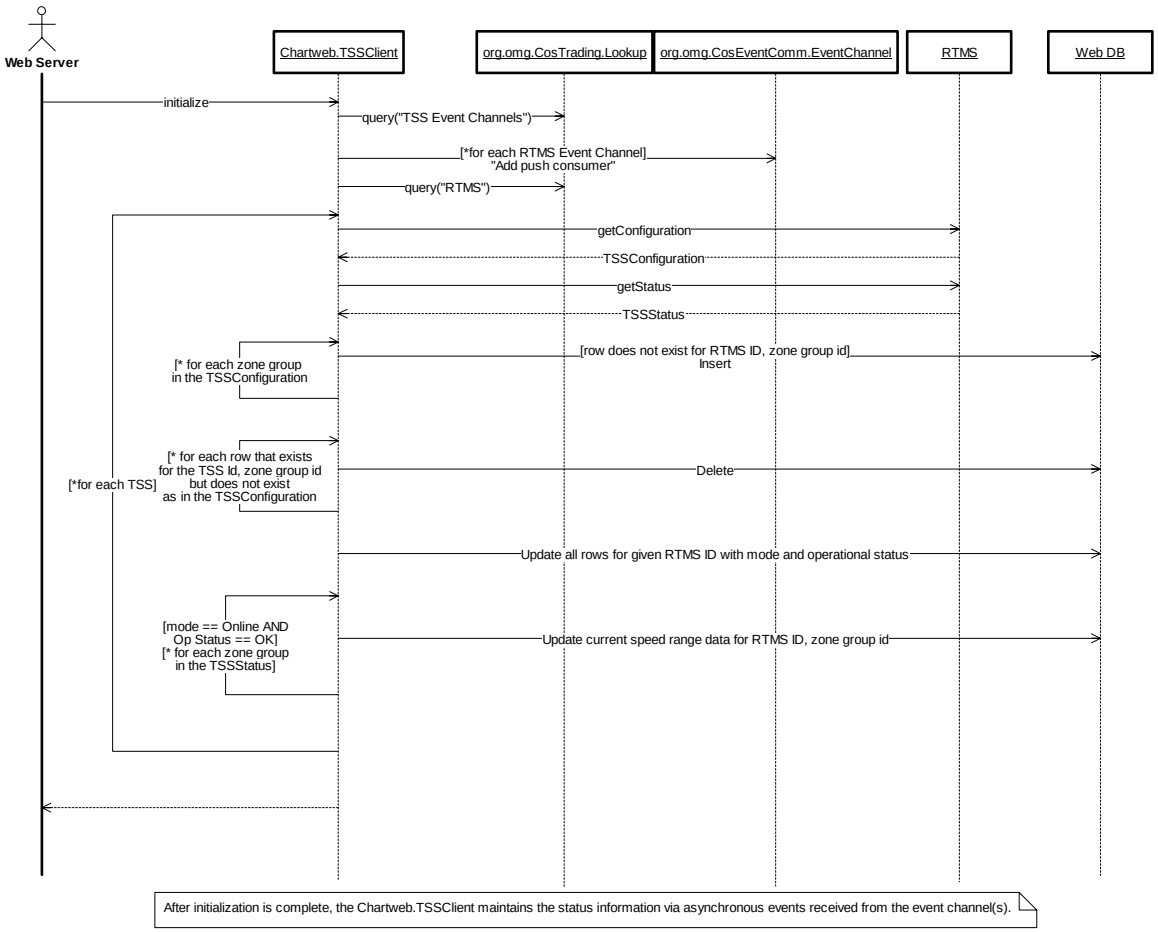

Figure 16. SummarizeRoadwayStatus:Initialize (Sequence Diagram)

3.3.15 SummarizeRoadwayStatus:ModeChanged (Sequence Diagram)

The CHART II system pushes an event to the Chartweb.TSSClient when the mode (online, offline, or maintenance) for an RTMS is changed. The Chartweb.TSSClient handles this event by updating all rows in the web database for the given RTMS ID with the new mode. The web map will only show data for rows in the database that are marked "online."

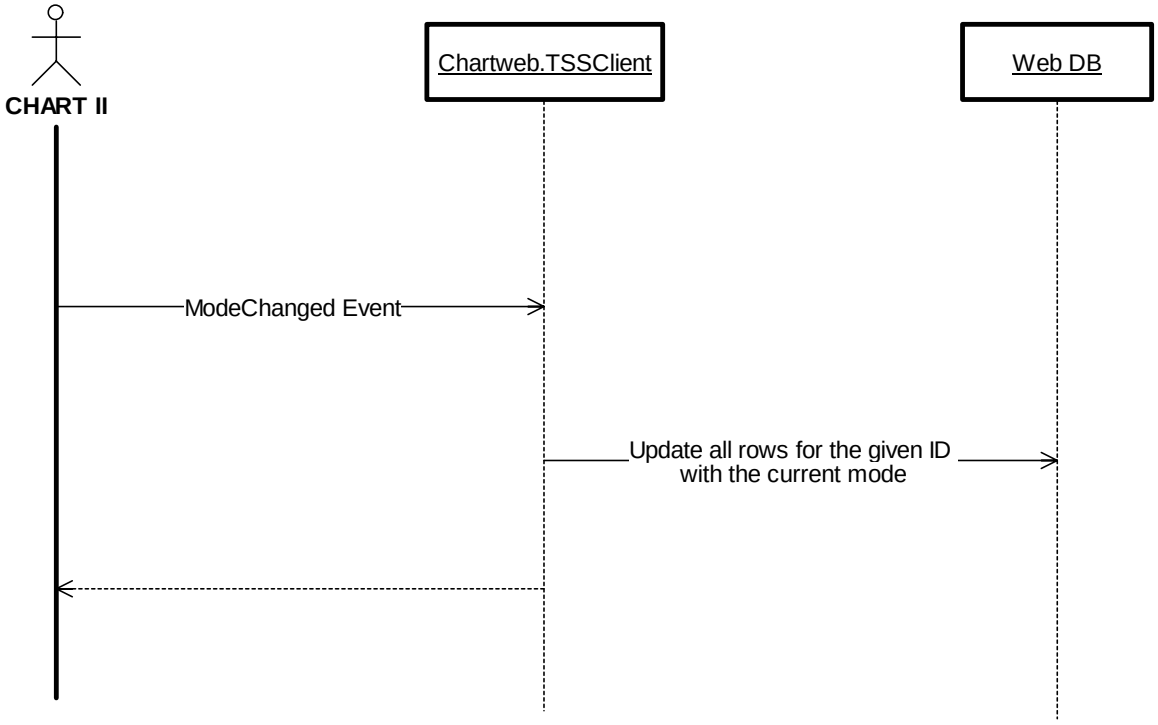

Figure 17. SummarizeRoadwayStatus:ModeChanged (Sequence Diagram)

3.3.16 SummarizeRoadwayStatus:ObjectAdded (Sequence Diagram)

The CHART II system pushes an event to the Chartweb.TSSClient when an RTMS is added to the CHART II system. The Chartweb.TSSClient handles this event by inserting a new row in the web database for each zone group configured for the RTMS (if a row with the given key does not already exist).

After rows are added to the web database, configuration of the web map must be done to associate map symbols with each row in the database. Note that all known RTMS objects can be added to the CHART II system during the initial deployment, even though the RTMS devices may not be functioning (They are added in an offline state). This would allow an initial population of the CHART II system and the Web Map to be done on the initial deployment. Using this approach, as each RTMS is brought online, its data will automatically appear on the Web Map without further configuration.

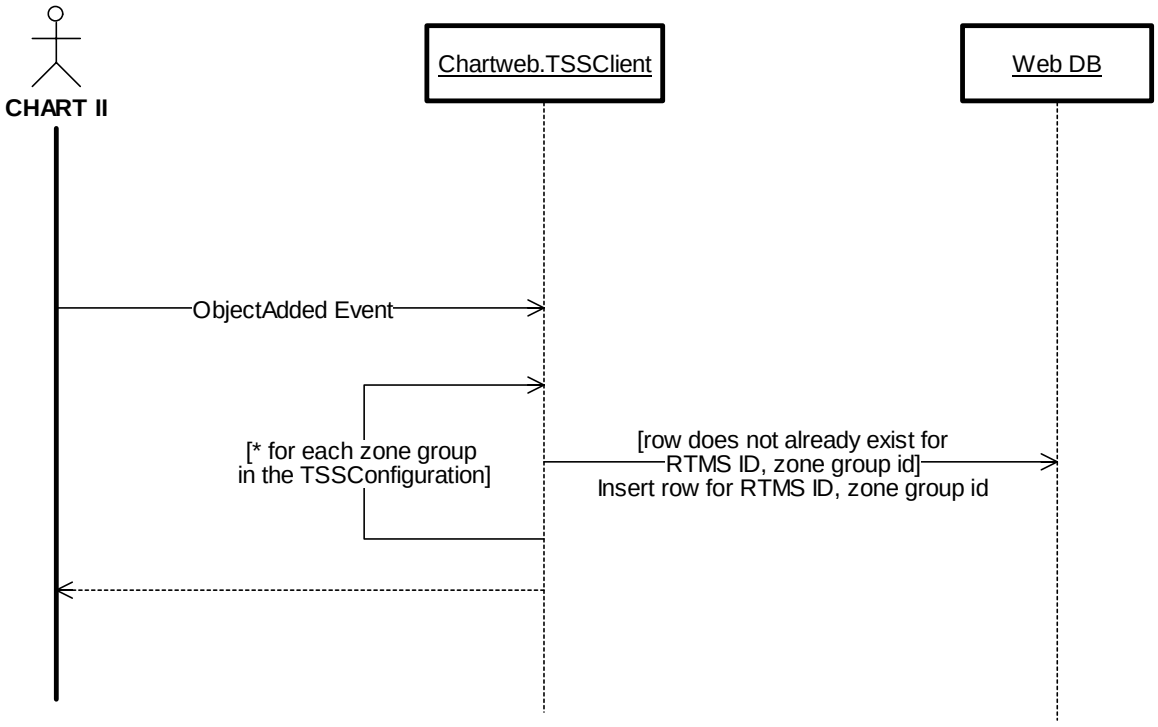

Figure 18. SummarizeRoadwayStatus:ObjectAdded (Sequence Diagram)

3.3.17 SummarizeRoadwayStatus:ObjectRemoved (Sequence Diagram)

The CHART II system pushes an event to the Chartweb.RTMSClient when an RTMS is removed from the CHART II system. The Chartweb.RTMSClient handles this event by deleting all rows from the web database for the given RTMS ID.

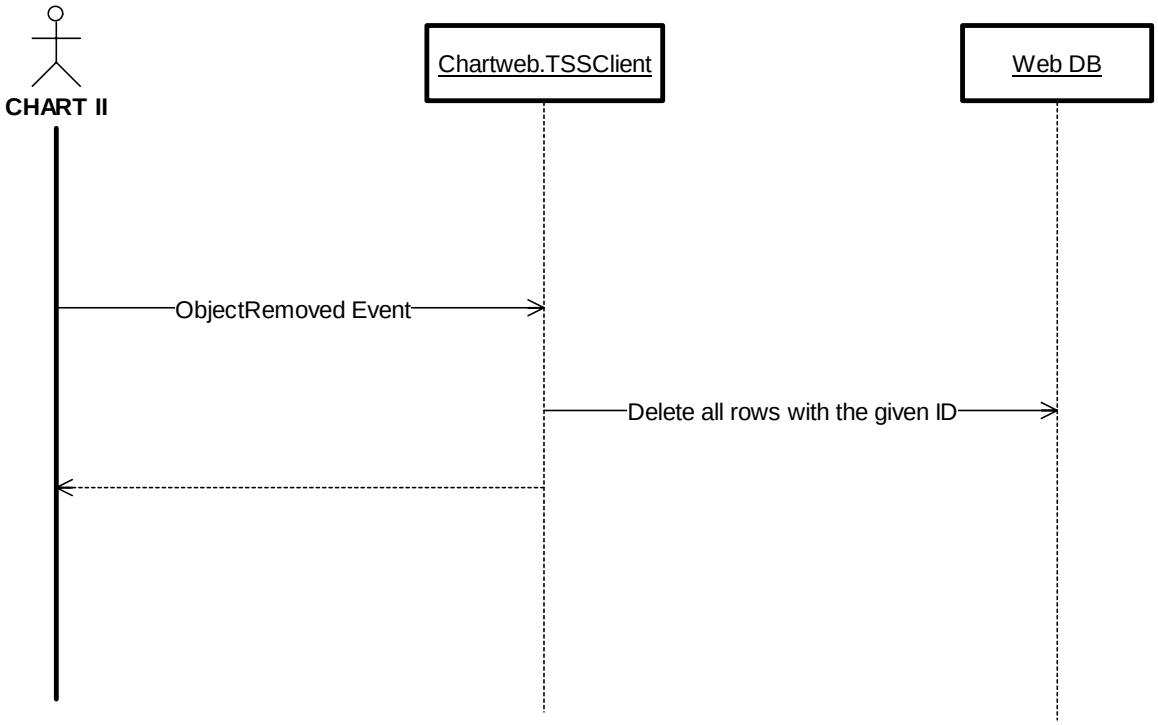

Figure 19. SummarizeRoadwayStatus:ObjectRemoved (Sequence Diagram)

3.3.18 SummarizeRoadwayStatus:OpStatusChanged (Sequence Diagram)

The CHART II system pushes an event to the Chartweb.TSSClient when the operational status (OK, Communications Failure, or Hardware Failure) for an RTMS has changed. The Chartweb.TSSClient handles this event by updating all rows in the web database for the given ID with the new operational status. The web map will only show data for rows in the database that are marked with an operational status of "OK."

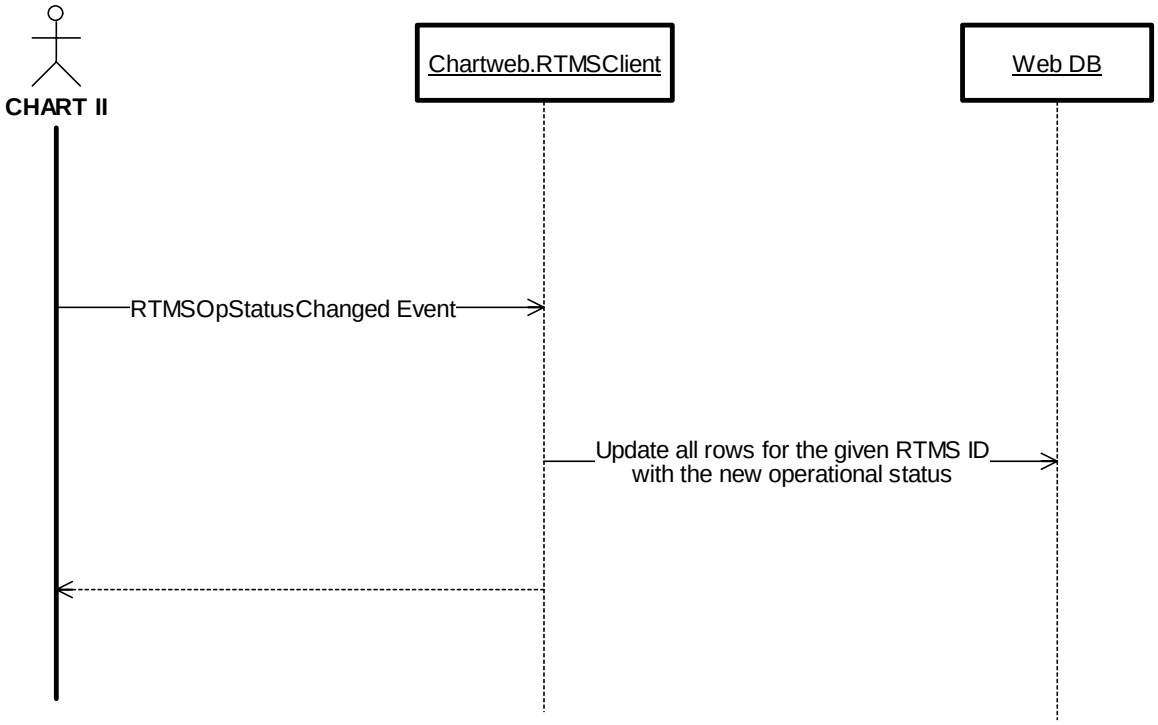

Figure 20. SummarizeRoadwayStatus:OpStatusChanged (Sequence Diagram)

3.3.19 TakeRTMSOffline:Basic (Sequence Diagram)

An administrator can take an RTMS offline if the RTMS is currently in maintenance mode or online. Polling of the RTMS is disabled when it is offline. An event is pushed to notify other applications and objects of the mode change.

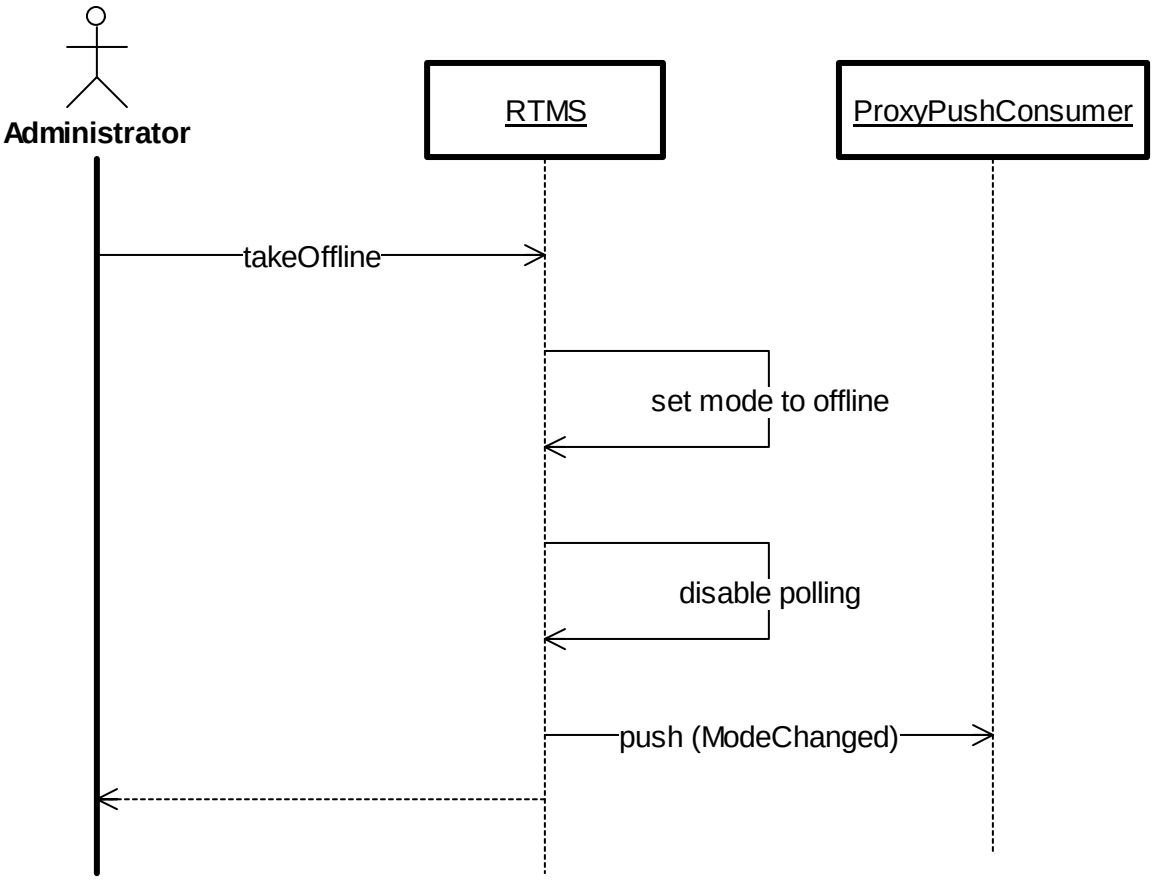

Figure 21. TakeRTMSOffline:Basic (Sequence Diagram)

3.3.20 ViewRoadwayStatus:Basic (Sequence Diagram)

When a web user requests the web page that contains the current speed data, the web server forwards a request to the map server to have it generate a map based on the current speed range data. The map server uses the speed range data from the web database when generating the map. The map server then responds to the web server, allowing the web server to serve the generated map to the web user.

The speed range data in the web database is kept up to date through a separate process. See the SummarizeRoadwayStatus.Basic sequence diagram for details.

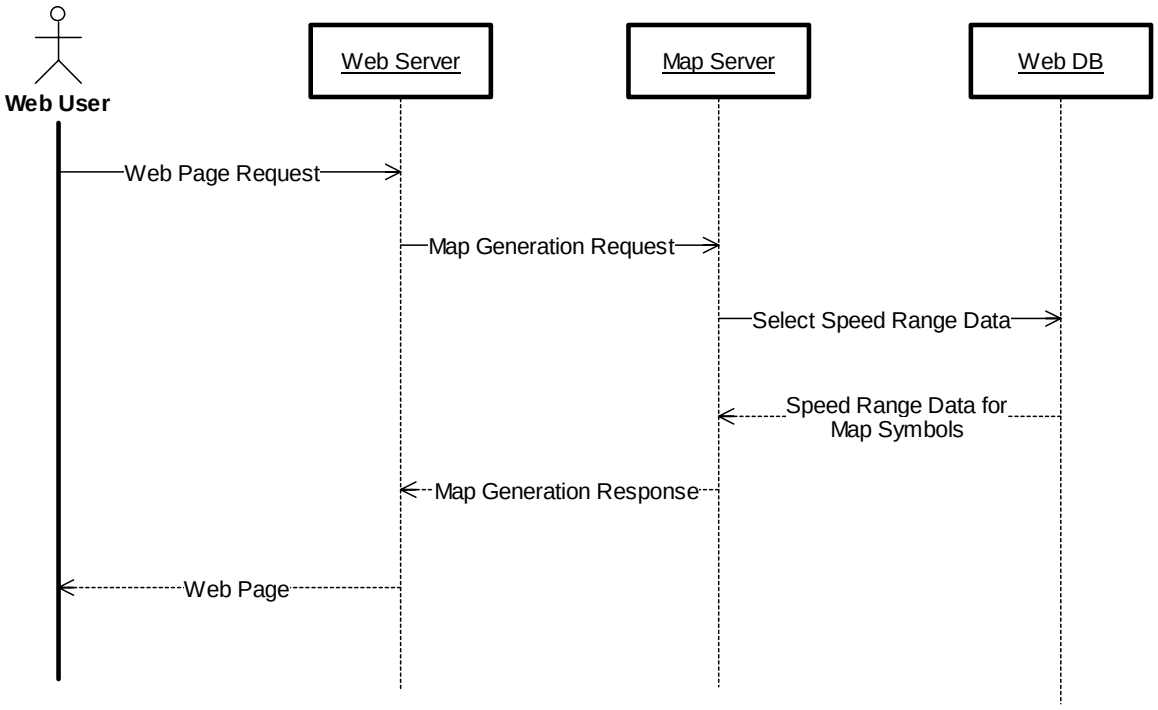

Figure 22. ViewRoadwayStatus:Basic (Sequence Diagram)

4 Packaging

4.1.1 TSSPackaging (Class Diagram)

This diagram shows the software packages involved in supplying RTMS data to the Web and the dependencies between the packages. The org.omg.* packages are provided by the ORB vendor. The CHART II team supplies the CHART2 packages. The Chartweb package represents the interface to be provided by the Web team to access the RTMS data (or any other Transportation Sensor System's data) from CHART II.

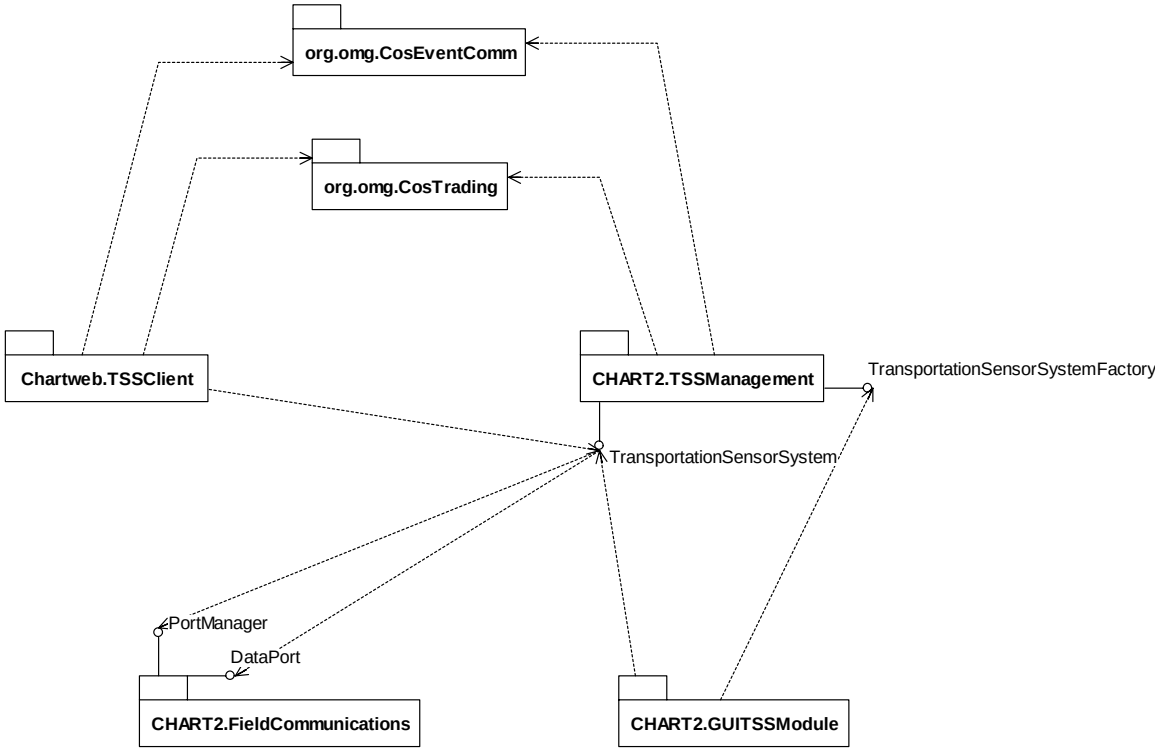

Figure 23. TSSPackaging (Class Diagram)

5 Deployment

5.1 RTMSDeployment (Deployment Diagram)

This diagram shows a representation of the types of server and client machines that exist within the system and the connections that exist between them.

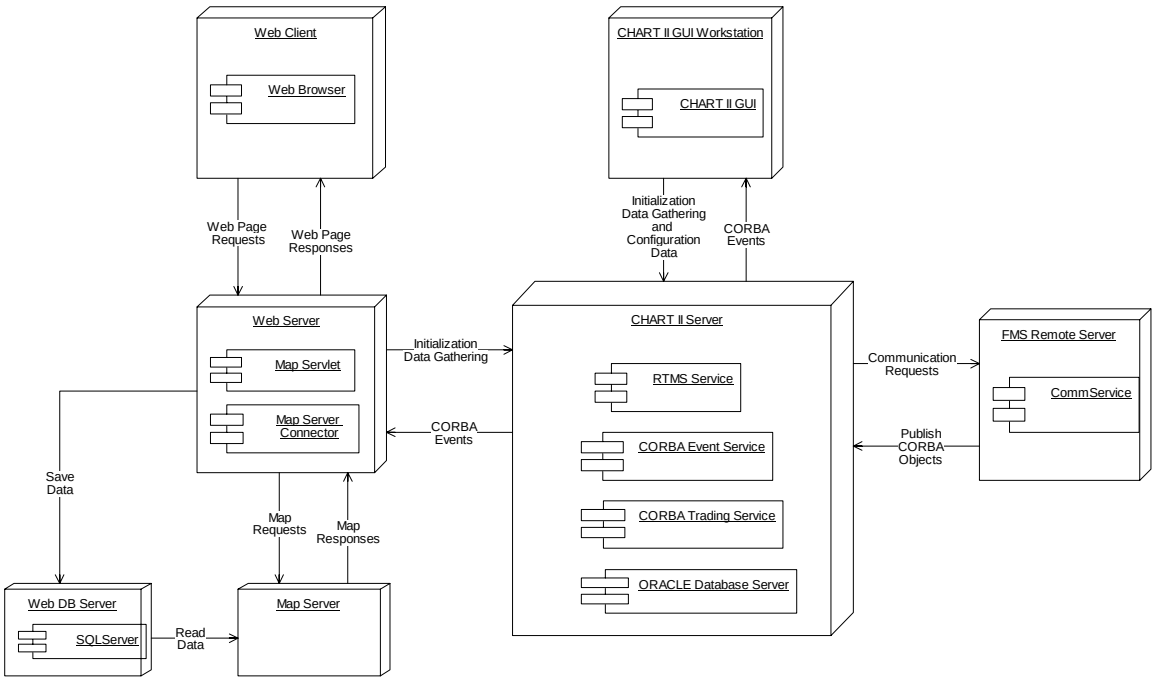

Figure 24. RTMSDeployment (Deployment Diagram)

Bibliography

CHART II Business Area Architecture Report, document no. M361-BA-005R0, Computer Sciences Corporation and PB Farradyne, Inc., April 28, 2000.

CHART II System Requirements Specification Release 1 Build 2, document no. M361-RS-002R1, Computer Sciences Corporation and PB Farradyne, Inc.

The Common Object Request Broker: Architecture and Specification, Revision 2.3.1, OMG Document 99-10-07.

FMS R1B2 High Level Design, document no. M303-RS-002R0, Computer Sciences Corporation and PB Farradyne, Inc., June 9, 2000.

Martin Fowler and Kendall Scott, *UML Distilled*, Addison-Wesley, 1997.

National Transportation Communications for ITS Protocol (NTCIP) Object Definitions for Transportation Sensor Systems (TSS) User Comment Draft, document no. NTCIP 1209 v01.09, AASHTO et al, July 12, 1999.

RTMS Coordination Meeting Minutes, document no. M361-MM-021R0, Computer Sciences Corporation and PB Farradyne, Inc., September 20, 2000.

RTMS Design Review Meeting Minutes, document no. M361-MM-023R0, Computer Sciences Corporation and PB Farradyne, Inc., October 12, 2000.

RTMS Requirements Review Meeting Minutes, document no. M361-MM-019R0, Computer Sciences Corporation and PB Farradyne, Inc., September 8, 2000.

Acronyms

The following acronyms appear throughout this document:

CHART	Coordinated Highways Action Response Team
CORBA	Common Object Request Broker Architecture
DBMS	Database Management System
FMS	Field Management Station
GUI	Graphical User Interface
IDL	Interface Definition Language
ISDN	Integrated Services Digital Network
ITS	Intelligent Transportation Systems
NTCIP	National Transportation Communications for ITS Protocol
OMG	Object Management Group
ORB	Object Request Broker
PC	Personal Computer
POTS	Plain Old Telephone System
RTMS	Remote Traffic Microwave Sensor
R1B2A	Release 1, Build 2A
TSS	Transportation Sensor System
UML	Unified Modeling Language

Glossary

Communications Server	A PC outfitted with communications hardware and the FMS Communications Service software.
Graphical User Interface	Part of a software application that provides a graphical interface to its user.
Occupancy	A measure of the usage of a roadway's capacity, expressed as a percentage.
Operator	A Chart II user that works at an Operations Center.
Port	A software object used to model a physical communications port.
Port Manager	A software object that manages access to one or more communications ports.
Protocol Handler	A software object that contains code that encapsulates the specific communications sequences required to command a field device.
RTMS	A sensor used to detect the volume, speed, and occupancy of traffic at a location on the highway.
Transportation Sensor System	A system capable of sensing and communicating traffic parameters.
User	A user is someone who uses the CHART II system. Users can perform different operations in the system depending upon the roles they have been granted.
Volume	A measure of the number of vehicles that have traveled across a section of roadway during a sample period.

Appendix A: CORBA Information

CORBA

CORBA is an architecture specified by the Object Management Group (OMG) for distributed object oriented systems. The CORBA specification provides a language and platform independent way for object oriented client/server applications to interact. The CORBA specification includes an Object Request Broker (ORB), which is the middleware used to allow client/server relationships between objects. Using a vendor's implementation of an OMG ORB, software applications can transparently interact with software objects anywhere on the network without the application having to know the details of the network communications.

Interfaces to objects deployed in a CORBA system are specified using OMG Interface Definition Language (IDL). Applications written in a variety of languages or deployed on a variety of computing platforms can use the IDL to interact with the object, regardless of the language or computing platform used to implement the object.

CORBA Services

The OMG CORBA specification includes specifications for application servers that provide basic functionality that is commonly needed by distributed object systems. While there are specifications for many such services, many services have not yet been implemented. Of the CORBA Services that are available, the CORBA Event Service and CORBA Trading Service are utilized in the CHART II system. A description of each of these services follows.

CORBA Event Service

The CORBA Event Service provides for a way to provide data updates within the system in a loosely coupled fashion. This loose coupling allows applications with data to share to pass the information via the event service without needing to have knowledge of others that are consuming the data.

Data passed through the event service is done using event channels. Many different types of events may be passed on a single event channel. Interested parties may become consumers on a given event channel and receive all events passed on the channel.

The CHART II system makes use of multiple event channels to allow event consumers to be more selective about the type of events they receive. Also, event channels of the same type may exist in multiple regions, allowing the CHART II system to be expandable and multi-regional. Event channels used in the CHART II system are published in the CORBA trading service to allow others to select which events they wish to consume.

CORBA Trading Service

The CORBA Trading Service is an online database of objects that exist in a distributed object system. Servers that have services to offer publish their objects in the trading service.

Applications that wish to use the services provided by a server can query the Trading Service to find objects based on their type or attributes.

CORBA Trading Services can be linked together into a federation. Queries done on single Trading Service can be made to cascade to all linked Trading Services as well. This feature allows Trading Services serving single regions to be linked together, providing seamless access to all objects in the system.

The CHART II System utilizes the CORBA Trading Service to allow the GUI to discover objects in the system with which it allows the user to interact. Using the linking capabilities of the Trading Service, the CHART II system can be distributed to multiple districts with the GUI still able to provide a unified view of the system to the users.